NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

THE PERFORMANCE CHARACTERISTICS OF AN ELECTROOPTIC SNS ANALOG TO DIGITAL CONVERTER

by

Hiromichi Yamakoshi

June, 1995

Thesis Advisor: Phillip E. Pace
Thesis Co-Advisor: Ronald J. Pieper

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 15 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE
THE PERFORMANCE CHARACTERISTICS OF AN ELECTROOPTIC SNS ANALOG TO DIGITAL CONVERTER

5. FUNDING NUMBERS

6. AUTHOR(S) Yamakoshi, Hiromichi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

High speed data conversion is required to conduct digital signal processing on wideband analog waveforms. This thesis explores the performance characteristics of a high speed optical analog to digital conversion technique. This technique extends the resolution of an integrated optical multi-interferometer analog to digital converter (ADC) by using a symmetrical number system (SNS) encoding. The optical SNS ADC can directly digitize wideband signals with a minimum amount of circuit complexity. Using an 8 bit SNS ADC, the results are significantly improved with a modification which incorporates a parity circuit, and a pulse timing circuit. The parity circuit decimates the possible encoding errors that can result when the samples lie at one of the code transition points. The results of this decimation process, and the timing of the corresponding signals are also discussed. Other areas such as the integration of the timing processor and folding circuit, are identified as those in which further design is required.

14. SUBJECT TERMS
Analog-to-digital converter (ADC), Symmetrical Numbering System (SNS), Mach-Zehnder Interferometer(MZI)

15. NUMBER OF PAGES * 131

16. PRICE CODE

17. SECURITY CLASSIFI-CATION OF REPORT	18. SECURITY CLASSIFI-CATION OF THIS PAGE	19. SECURITY CLASSIFI-CATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

THE PERFORMANCE CHARACTERISTICS OF AN ELECTROOPTIC SNS ANALOG TO DIGITAL CONVERTER

Hiromichi Yamakoshi
Lieutenant, Japan Maritime Self Defense Force
B.S.Equivalent, Japan National Defense Academy, 1988

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1995

Author: Hiromichi Yamakoshi
Hiromichi Yamakoshi

Approved by:
Phillip E. Pace, Thesis Advisor

Ronald J. Pieper, Thesis Co-Advisor

Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

High speed data conversion is required to conduct digital signal processing on wideband analog waveforms. This thesis explores the performance characteristics of a high speed optical analog to digital conversion technique. This technique extends the resolution of an integrated optical multi-interferometer analog to digital converter (ADC) by using a symmetrical number system (SNS) encoding. The optical SNS ADC can directly digitize wideband signals with a minimum amount of circuit complexity. Using an 8 bit SNS ADC, the results are significantly improved with a modification which incorporates a parity circuit, and a pulse timing circuit. The parity circuit decimates the possible encoding errors that can result when the samples lie at one of the code transition points. The results of this decimation process, and the timing of the corresponding signals are also discussed. Other areas such as the integration of the timing processor and folding circuit, are identified as those in which further design is required.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

I would like to acknowledge the guidance of Professor Phillip E. Pace, my thesis advisor, and Professor Ronald J. Pieper, my thesis Co-advisor. I would also like to thank Professor John P. Powers, and Mr. Rick Patterson from the Naval Research and Development Division of the Navy Command and Control Ocean Surveillance Center for their suggestions on circuit design. CAPT. Charles Ristorcelli, USN and Mr. Jerry Peake from the Space and Naval Warfare System Command provided invaluable support, and LT. Rickey D. Walley, my project partner, gave me a lot of helpful guidance during this research effort. CDR. Robert Glenn Handlers and LT. Mary Ellen Green also deserve recognition with helping me improve my research and thesis writing skills. Finally, I would like to thank the Japan Maritime Self Defence Force for giving me the opportunity to be intellectually challenged at the Naval Postgraduate School, and my parents who encouraged me often from overseas.

I. INTRODUCTION

A. OVERVIEW OF ANALOG-TO-DIGITAL CONVERTER

1. Classification of Analog to Digital Converters

Analog to digital conversion is a necessary process in many digital electronic circuits. This process converts an analog signal to an accurate digital number proportional to the input amplitude. We will refer to devices converting analog signals to digital representations as analog-to-digital converters (ADCs), and those which perform the reverse transformation (digital inputs to proportional analog signals) as digital-to-analog converters (DACs). Pulse-code modulation (PCM), measurement instruments (multimeter, digital memory oscilloscopes for example), signal-generation and processing instruments (waveform synthesizer for example) are all common applications that use analog to digital conversion.[Ref. 1: pp. 612-614]

There are four main types of ADCs: counter (or ramp) ADC, successive approximation ADC, flash (or parallel) ADC, and integrating ADC.[Ref. 2: pp. 550-561] Each type has its advantages and disadvantages, and we choose one of these types based on our application. The counter ADC is composed of four parts: the comparator, timing clock, digital counter, and DAC. Figure 1 shows a block diagram of a counter ADC. We assume that the initial setting of the counter is zero. When an analog sample is input to the comparator, the counting process is initiated. The comparator output goes into the digital counter with the clock pulse gate. Each succeeding clock pulse then increments the counter's binary sequence output. The DAC translates this binary code to an analog signal level. The comparator then compares the DAC output to the input analog sample. When the DAC output exceeds the level of the analog sample, the comparator output changes states, stopping the digital counter. The final counter state is the desired digital representation of the analog sample. The counter ADC is the simplest type to understand. However, it is relatively slow compared to the others because of the many comparisons which must be performed at each step. A counter ADC with an n-bit output may require as many as $2^n - 1$ comparisons for each input sample.

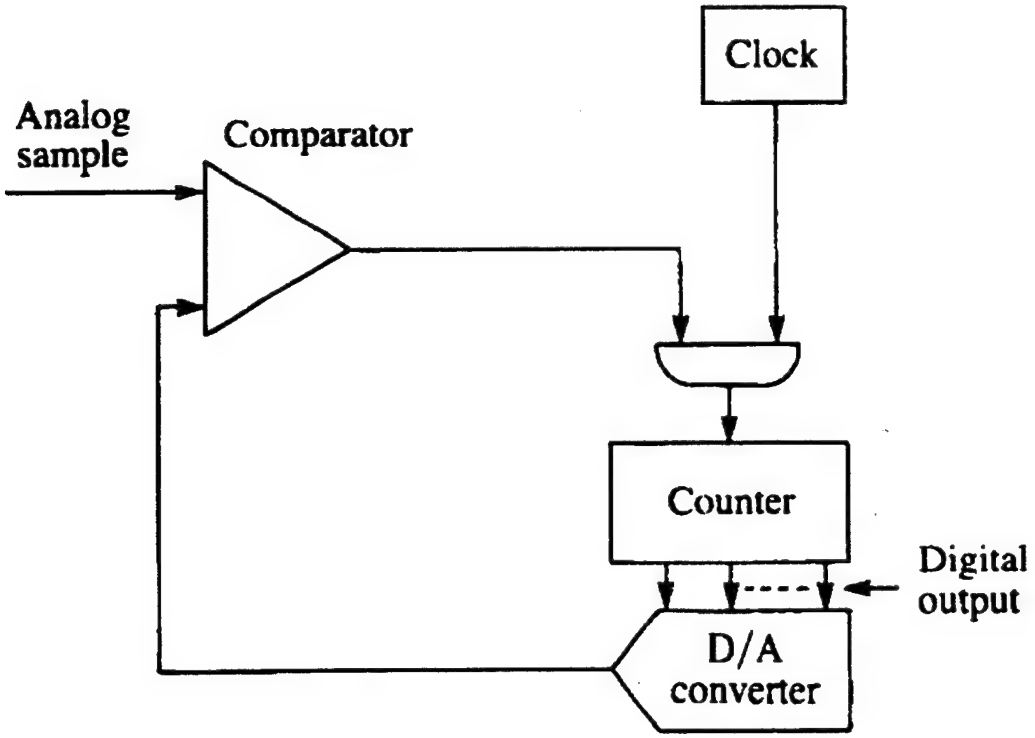


Figure 1. Block Diagram of a Counter ADC from Ref. [2].

Successive approximation is another common general purpose ADC type. Figure 2 shows the block diagram of a successive approximation ADC. This type uses control logic and an output register, referred to as a successive approximation register (SAR), in place of the clock pulse and counter of the counter ADC. The comparator compares the output of the DAC to the input analog sample. The completion of conversion is triggered by a change in the state of the comparator. The diagram of both the counter ADC and the successive approximation ADC is similar, however, the logic between both ADCs is different.

The flow chart for a successive approximation ADC is shown in Fig. 3. While the counter ADC compares the next state of converted-analog signal with the reference signal, the successive approximation ADC compares the bit information from the most significant bit (MSB) to the least significant bit (LSB). The successive approximation ADC thus requires at most n comparisons, so the conversion time is clearly much shorter. A typical conversion time of the successive approximation ADC ranges from 1 μsec to 50 μsec.

The flash ADC has the shortest conversion time of any available type of ADC. The block diagram of a flash ADC is shown in Fig. 4. An n-bit flash ADC requires $2^n - 1$ comparators. The input analog sample is applied simultaneously to all comparators. The reference voltage for each comparator is set through a common resistor ladder. The decoder converts the comparator output pattern to a digital word representing the analog sample. For example, if we have seven comparators in the ladder and three of the lower comparator outputs are set to the "ON" state, then the ladder creates a digital word "0000111." This digital word representation is often called a thermometer code. The decoder converts this code to a common binary representation "011." For a decoder circuit, a read only memory (ROM) is widely used. Because of using $2^n - 1$ comparators, we do not need to set up a feedback loop. We need only one comparison of the analog sample to get the desired digital representation. The flash ADC is, needless to say, the fastest, but also the most expensive ADC. In addition, the number of bits and the practical resolution are limited because of the many comparators loaded in parallel.

The integrating ADC is used for special applications. This type uses input signal integration, which may improve the signal-to-noise ratio (SNR) for certain types of signals in the circuit.[Ref. 2: pp. 557-561] Thus, using an integrating ADC may help suppress the

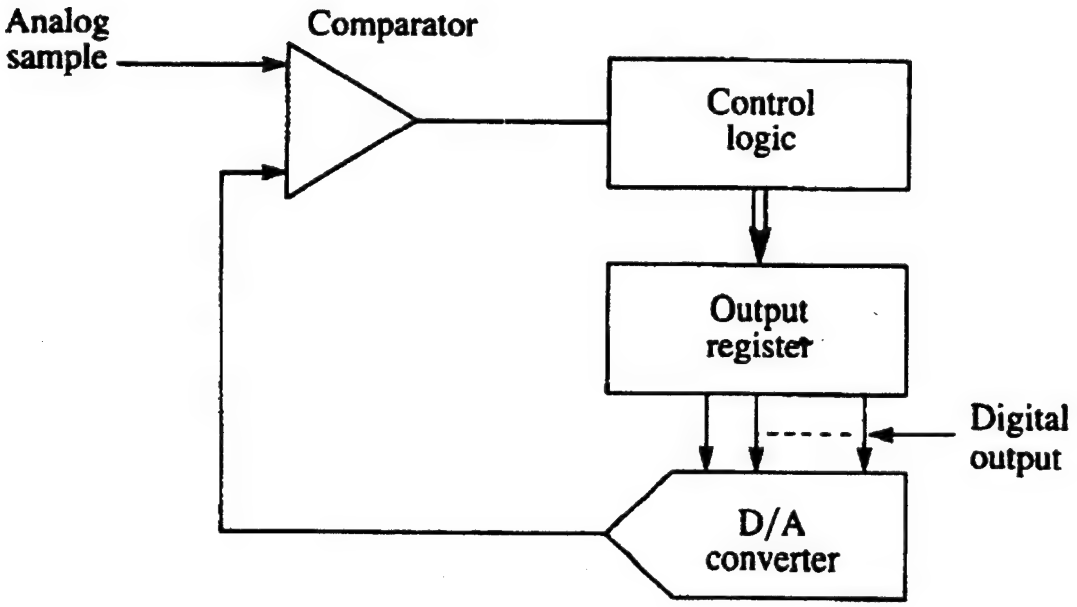


Figure 2. Block Diagram of a Successive Approximation ADC from Ref. [2].

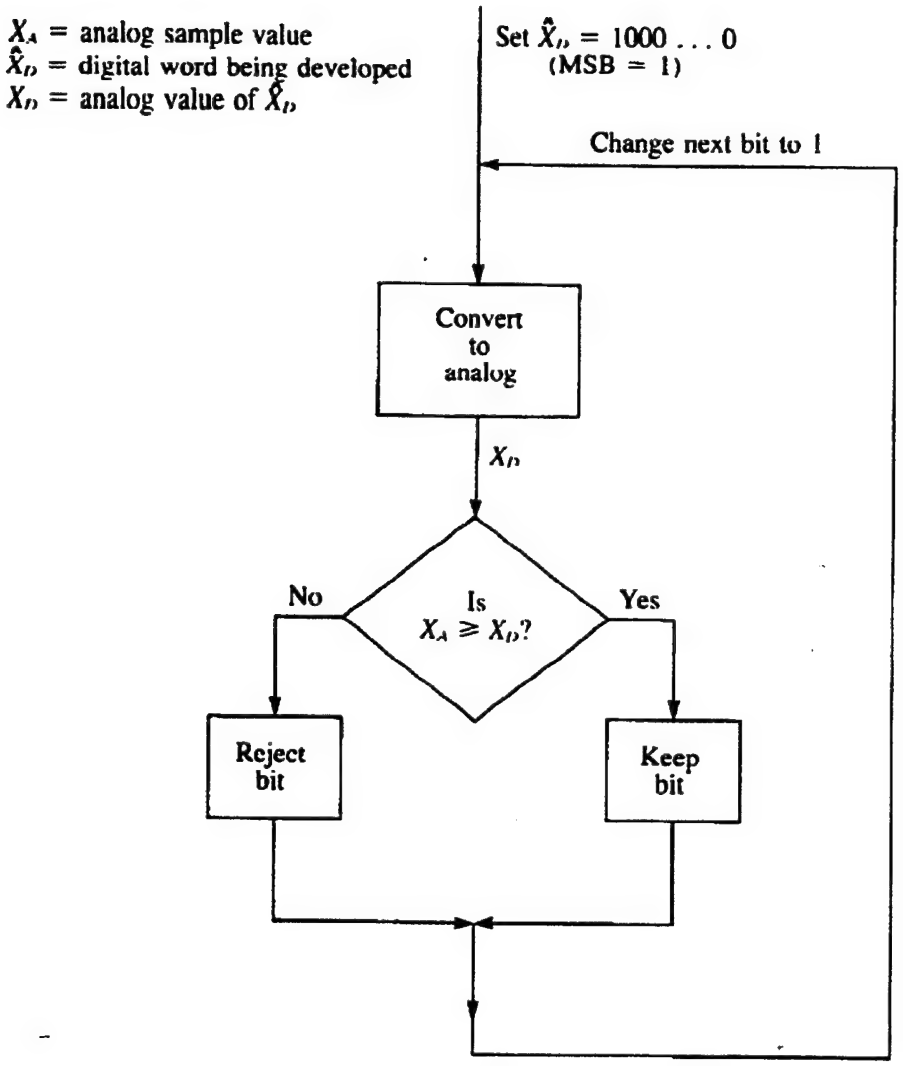


Figure 3. Flow Chart for a Successive Approximation ADC from Ref.[2].

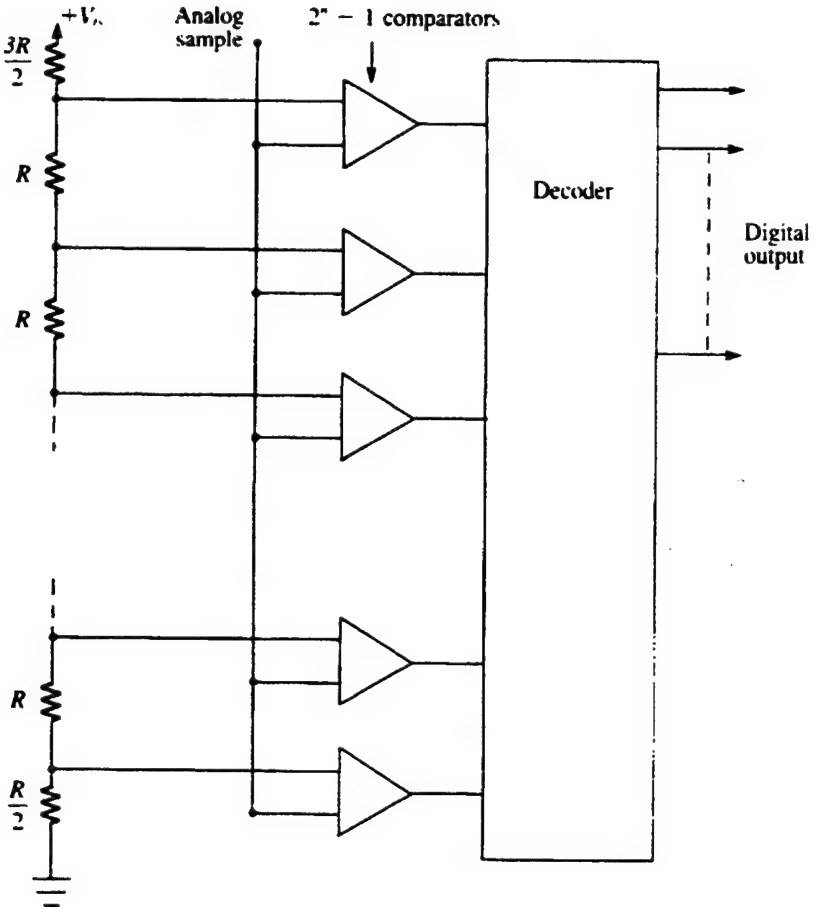


Figure 4. Block Diagram of a Flash (Parallel) ADC from Ref. [2].

input noise, but the circuit's frequency response curve distorts the spectrum of the signal. Therefore, the integrating ADC is used only for dc and low frequency (slowly varying) signals to preserve the integrity of the signal. Also, this type of ADC is relatively slow due to the precision integrator circuits. For example, with the Delta-Sigma ADC (Δ-Σ ADC) which uses the integrating ADC scheme, the conversion rates are up to 300 msec.[Ref. 3] The block diagram of a Δ-Σ ADC is shown in Fig. 5. The input voltage of the Δ-Σ ADC drives the integrator and is compared to a fixed voltage range such as ground. Depending on the comparator output, fixed elements of charge are switched into the summing junction. A counter keeps track of number of charge pulses switched into the summing junction, and the output is a digital word.

2. Current Trend of Developing ADCs

ADCs are widely used to translate an analog signal input into a digital signal output. In fields such as digital computing, control systems, and information processing, ADCs play a key role. Recent advancements in scientific technology have provided for a shorter timing response in signal processing therefore, faster sampling rates by the ADC are required for these high speed applications. As mentioned in the previous section, the flash ADC has the fastest response of any of the current ADCs. Some research papers indicate an effort to improve the sampling rate and resolution using electronic folding devices.[Ref. 4-6] In these converters, the input signal is folded in parallel in an attempt to reduce the complexity of the amplitude analyzing function. One of the newest high speed folding ADC architectures is the integrated optical guided wave (IOGW) technique.[Ref. 7-13] The IOGW technique can handle higher sampling rates than conventional ADCs and can be used for high speed applications.

B. INTEGRATED OPTICAL SNS ADCS

1. Symmetrical Number System

The symmetrical number system (SNS) is composed of a number of pairwise relatively prime (PRP) moduli m_i. The integers within each SNS modulus are representative of a symmetrically folded waveform with the period of the waveform equal to twice the PRP modulus. For m given, the integer values within twice the individual modulus x_m are given by:

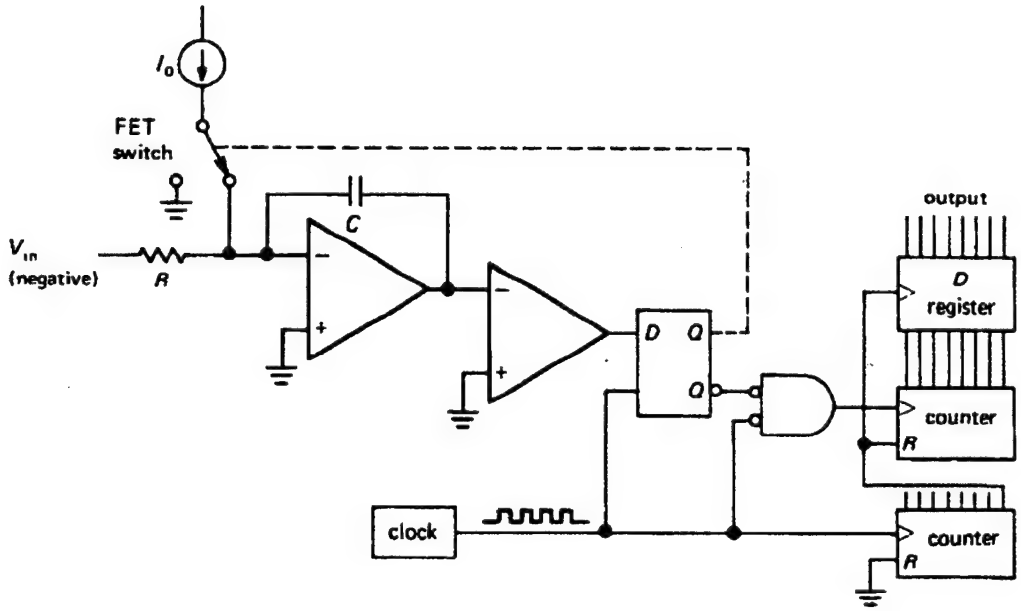


Figure 5. Block Diagram of a Δ–Σ ADC from Ref. [1]

$$x_m = [0,1,\ldots,m-1,m-1,\ldots,1,0]. \quad (1)$$

The SNS folding waveform is symmetrical about the midpoint and is compatible with other folding waveforms. Due to the presence of ambiguities, the integers within Eq. 1 do not form a complete system of length $2m$ by themselves. It is well known however, that the inclusion of additional redundant moduli can effectively detect and correct errors within a residue number system (RNS) representation of a number. The SNS formulation is based on a similar concept which allows the ambiguities to occur. The ambiguities that arise within the SNS are resolved by using various arrangements of the SNS moduli. By considering the derived moduli arrangements, the SNS is rendered a complete system having a one-to-one correspondence with the RNS. For N equal to the number of PRP moduli, the dynamic range M of the system is as follows [Ref. 11]:

$$M = \prod_{i=1}^{N} m_i \quad (2)$$

where M is also the position of the first repetitive moduli vector.

2. Integrated Optical SNS ADC

The most efficient (sampling) ADCs demonstrated to date use IOGW technology (in lithium niobate ($LiNbO_3$), gallium arsenide (GaAs), or indium phosphide (InP)). Symmetrical folding of the input signal is realized with Mach-Zehnder Interferometer (MZI) waveguide modulators arranged in a parallel configuration.[Ref. 7-10] These modulators use the linear Pockels effect, and provide a convenient method for coupling a wideband electrical signal into an optical processing system through the modulator electrodes. This architecture, often called a Taylor scheme, makes use of the periodic dependence of the modulator optical output on the applied voltage to the electrodes and the electrode lengths. The ultimate bandwidth of the signal that can be digitized is determined by the sampling pulse width and rate, the modulator bandwidth, the detector bandwidth, and the speed of the output digital circuitry. One of the major limitations for the Taylor scheme, however, is the required doubling of the electrode lengths for the increasing number of bits. For

example, an n-bit ADC using a Taylor scheme would require electrode lengths of l, $2l$, $4l$, ..., $2^{n-1} \cdot l$ where the length of the longest modulator is $2^{n-1} \cdot l$. Here l stands for the shortest electrode length. This electrode-length doubling requirement results in a high device capacitance, and ultimately constrains the achieved resolution to about 2 to 4 bits, which is not sufficient for many applications.[Ref. 10]

Recently, a technique that extends the resolution of an integrated optical multi-interferometer analog-to-digital converter was described.[Ref. 11-13] The optical output waveform for each interferometer is symmetrically folded at twice a proper modulus. A small comparator ladder mid-level quantizes each interferometer's detected output to encode the analog signal in a SNS format. By incorporating the SNS encoding, resolution greater than 1-bit per interferometer can be provided.

By applying Eq. 1 and Eq. 2, the SNS can serve as a source for resolution enhancement in an ADC by decomposing the analog amplitude analyzing function into a number of parallel sub-operations (moduli) that are of smaller computational complexity. Each sub-operation for a different modulus requires only a precision based on that modulus. A much higher resolution is achieved after the results of these low precision sub-operations are recombined. That is, the resolution of each interferometer is greater than 1-bit per interferometer. An original schematic diagram of a SNS ADC architecture is shown in Fig. 6. The input signal is folded in parallel with each folding period equal to twice a particular modulus. The folded waveform at the output of each folding circuit is mid-level quantized with a small comparator ladder to encode the input signal into the SNS format. An encoder then converts the SNS representation to a more familiar digital output such as a binary representation. With the SNS encoding any combination of folding periods and comparator arrangements, any analysis can be exact.

The optical output power from a single guided-wave interferometer is symmetrical and periodic, and can thus be used to implement each folding circuit (modulus) in the SNS ADC. The interferometer normalized output is a function of both the sampled analog voltage V and the modulus m_i as

$$I(v_i) = \sin^2\left(\frac{\pi v_i}{2V_\pi}\right) \qquad i \in \{1,2,\ldots,N\} \tag{3}$$

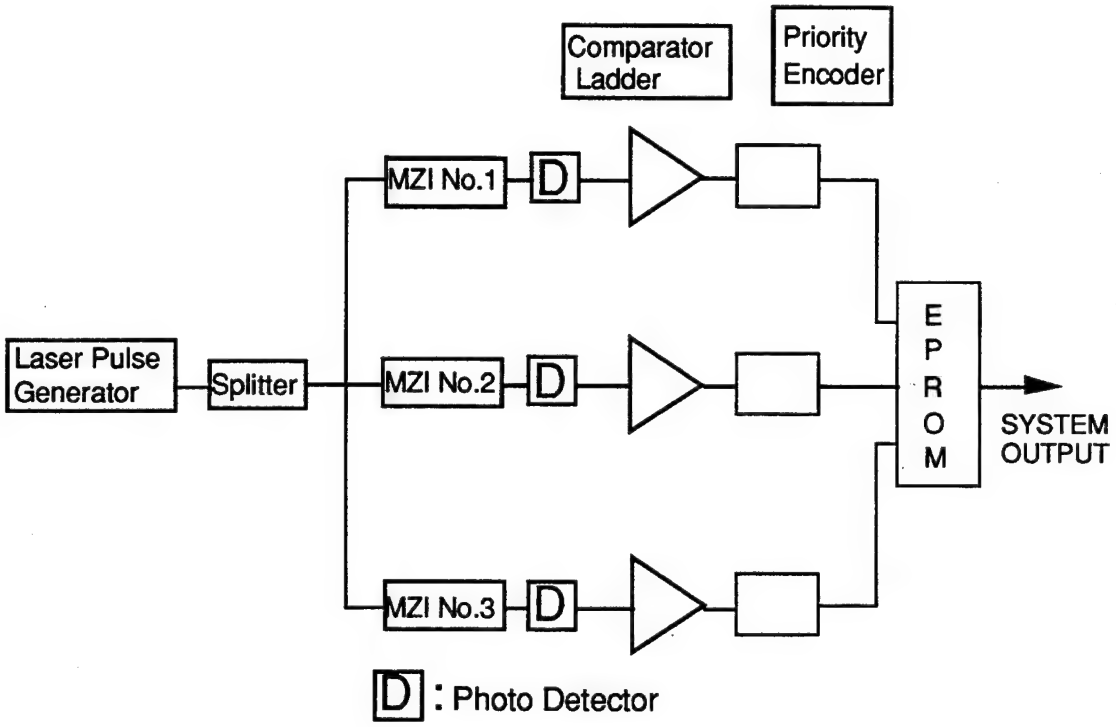


Figure 6. Original System Diagram of an SNS ADC

where v_i is the modulus dependent attenuated input signal, and N is the number of modulators. The attenuation factor depends on the modulus in such a way that identical interferometers can be used. Specifically, we can take

$$v_i = \frac{m_{min}}{m_i} V \leq V \tag{4}$$

where V is the input signal, and V_π is the half wave voltage for the MZI.[Ref. 14: pp. 324-330] The m_{min} is the minimum modulus of the set needed to satisfy Eq. 4.

The folding circuit for each modulus requires m_i - 1 comparators at the output of the detector. With the waveform given by Eq. 3, the normalized threshold values for the comparators within each modulus m_i are

$$T_{i,j} = \sin^2\left(\frac{\pi}{2} \cdot \frac{v_j}{V_\pi}\right) \tag{5}$$

where

$$v_j = \frac{j}{m_i} V_\pi \qquad j \in \{1,2,\ldots, m_i - 1\}. \tag{6}$$

That is, the comparator thresholds are tailored to the interferometer output waveform. The size of the LSB for the SNS ADC is obtained from Eq. 4 and Eq. 6, under the condition that $m_i = m_{min}$,

$$V_{LSB} = \frac{V_\pi}{m_i} \tag{7}$$

and therefore a larger minimum modulus leads to higher resolution.

Table 1 shows several possible SNS ADC configurations. Systems with resolution on the order of seven to eleven bits are shown. Detailed are the moduli corresponding to each interferometer with the dynamic range, the total number of comparators required, and the maximum number of comparators loaded in parallel at a detector output (fan out).

Another device constraint is the maximum voltage that may be applied to the electrodes. Applied bipolar voltage beyond the rated maximum (minimum) will spark across the electrode structure and damage the device. The specifications of V_π and v_{max} reveal the maximum number of folds available from the device. An input voltage which makes one complete fold is $2\,V_\pi$. The maximum number of folds F_{max} available from the device is therefore

$$F_{max} = \frac{2v_{max}}{2V_\pi} . \tag{8}$$

The smallest modulus within the SNS system requires the largest number of folds to instrument the dynamic range. Note that each fold in the minimum modulus covers $2m_{min}$ SNS ADC states. The largest number of folds required from an interferometer in a B-bit SNS ADC is therefore

$$F_{req} = \frac{2^B - 1}{2m_{min}} < \frac{v_{max}}{V_\pi} . \tag{9}$$

C PRINCIPAL CONTRIBUTIONS

In this thesis, MATLAB™ programming code of the SNS ADC is provided as a simulation of the theoretical system behavior. The initial program offers the normalized threshold level and the resistance value information in the ladder circuit of the SNS ADC, and the digital word conversion between the thermometer code and the binary code for an arbitrary modulus. Furthermore an 8-bit SNS ADC system simulation with a parity circuit

Interferometer Moduli	Dynamic Range M	Total Number of Comparators L	Max Number Loaded in Parallel K	Number of Bits B
3,4,5	60	9	4	5
11,12	132	21	11	7
5,7,8	280	17	7	8
9,10,11	990	27	10	9
13,14,15	2730	39	14	11

Table 1. Possible SNS ADC Configurations

to discard encoding errors is described. MATLAB is also used in this simulation. The program computes the normalized threshold levels of the parity circuit, the resistance values, and each resistance value in the ladder circuit. The parity circuit reduces the encoding error problems of the original SNS ADC, and this is measured using the logic analyzer. Comparison and discussion of the results between the actual SNS ADC output using LabVIEW generated inputs and the output computed by the MATLAB simulation is accomplished. Decimation using the parity circuit are also described.

Because of multi-interferometer operation, timing is needed to maintain proper operation of the SNS ADC. Therefore a timing processor is developed, and the circuit is constructed. Each component of the timing processor, and the timing characteristics of the processor are described.

D THESIS OUTLINE

Chapter II of this thesis consists of an overview of MATLAB followed by a description of the initial simulation development. Next, the extended development of the MATLAB program to include the parity analysis is discussed. Finally, total system simulation is obtained with MATLAB and documented. Chapter III provides the details on the timing processor of the system, and a description of the components considered. Chapter IV shows some experimental results of the actual circuit, and discusses the difference between the results simulated by MATLAB and the results of the actual circuits. Following the summary in Chapter V, the Appendix provides the MATLAB code that computes the design and simulates the whole SNS ADC system input. This input uses the same signal generating methods as LabVIEW. Each program in the Appendix is discussed in Chapter II.

II. SIMULATION OF THE SNS ADC USING MATLAB™

A. OVERVIEW

In optical integrated circuits, the linear electrooptic effect (or the Pockels effect) is widely used, especially for light modulation and switching. An index change is induced in the crystals via the electrooptic effect when an electric field is applied to the crystal. When the index varies linearly with the applied electric field, it is called the linear electrooptic effect. Since the linear electrooptic effect is found in crystals without an inversion symmetry, such as $LiNbO_3$, the sign of the induced index change depends on the polarity of the voltage applied to the crystal. The MZI is one of the popular optical computing devices that use the linear electrooptic effect to induce phase modulation in the optic signals. Equation 3 shows the output of the MZI which is a sinusoidal function. Each MZI folds the input signal before being digitized by high speed comparators.

The electronic part of the original SNS ADC is shown in Fig. 7. It includes a comparator ladder circuit, priority encoder, and EPROM (erasable programmable ROM). The EPROM is similar to the programmable logic array (PLA). The EPROM is a programmable and erasable device, which converts the thermometer code into a binary representation. In an ADC SNS design, the threshold voltage levels of the comparator ladder for each modulus need to be derived. However, the threshold voltage level for each modulus is different. One of the reasons for the different voltage levels is the non-linear characteristic of the modulator, and the differences in the folding periods. Also the threshold levels depend on each modulus in the SNS. According to the SNS structure, m-1 threshold values are needed for mod m and these threshold voltages can be expressed by applying Eq. 3. The characteristic curve of the modulator of each MZI is the same, and we can get the value of the predicted threshold voltage for each modulus using computer simulation.

MATLAB is a technical computing environment for high-performance numeric computation. It integrates numerical analysis and matrix operation, and its interface is very easy to use. MATLAB can also be configured for various computer platforms. The MATLAB program developed in this thesis is discussed in the following section. MATLAB is a trademark of The Math Works, Inc.

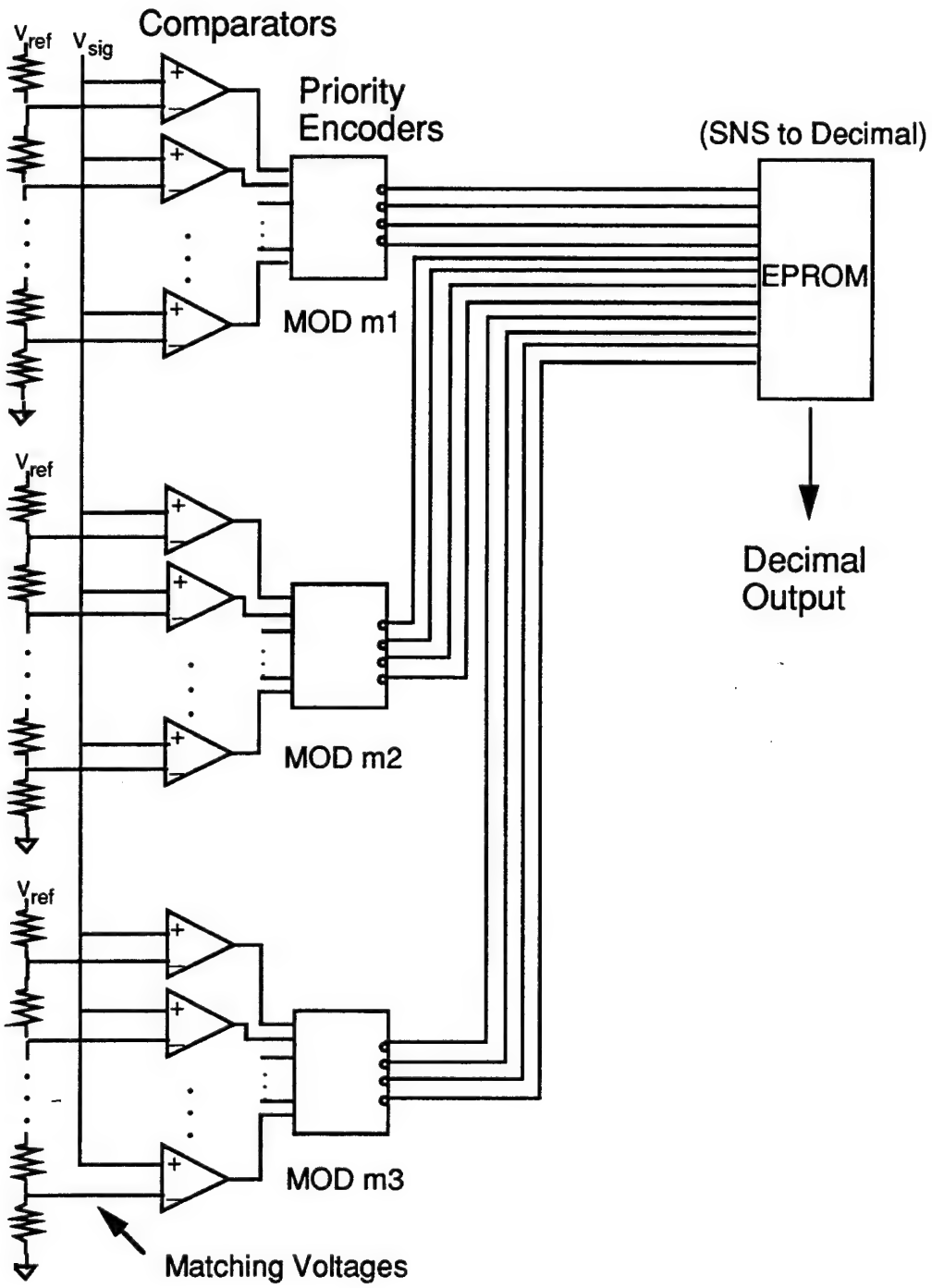


Figure 7. The Electronic Part of the SNS ADC

B. DEVELOPMENT OF MATLAB PROGRAM

1. Computing the Threshold Values for Each Modulus

The program to calculate the normalized threshold voltages (MATLAB code) is shown in Appendix A. The normalized value of the threshold voltage for each comparator in mod m can be obtained using the program called Vth_calc.m. This program applies Eq. 3, which details the relationship between the intensity of the MZI output and its corresponding input voltage. The m-1 threshold values in mod m are calculated, and each threshold value depends on the input voltage to the SNS ADC (D). An example drawing showing the threshold voltages (mod 9) is shown in Fig. 8. In this case, eight threshold values are needed and each value can be calculated using the first half period of the MZI output. The small circles on the graph show the threshold voltage levels for each input voltage index. The x-axis shows the index number for the input voltage D, and the y-axis shows the normalized intensity of the MZI output. Actually, we cannot measure the light intensity directly, but we can measure the voltage as an intensity related value using a photo detector. If we know the maximum voltage of the MZI output $\max(V_{MZI\,out})$, we can get each threshold voltage as;

$$Vth_{m_{ij}} = \max(V_{MZI\,out}) \cdot T_{i,j} \qquad \begin{matrix} i \in \{1,2,...,N\} \\ j \in \{1,2,...,m-1\} \end{matrix} \tag{10}$$

The Vth_calc.m program can also be modified for the general case, in which the minimum intensity of MZI output is not equal to zero. In this case, the threshold voltage can be calculated using the minimum and maximum voltage of the photo detector output:

$$Vth_{m_{ij}} = \min(V_{MZI\,out}) + (\max(V_{MZI\,out}) - \min(V_{MZI\,out})) \cdot T_{i,j} \qquad \begin{matrix} i \in \{1,2,...,N\} \\ j \in \{1,2,...,\mathrm{m}-1\} \end{matrix} \tag{11}$$

The SNS ADC uses multiple interferometers, and therefore calculating the threshold

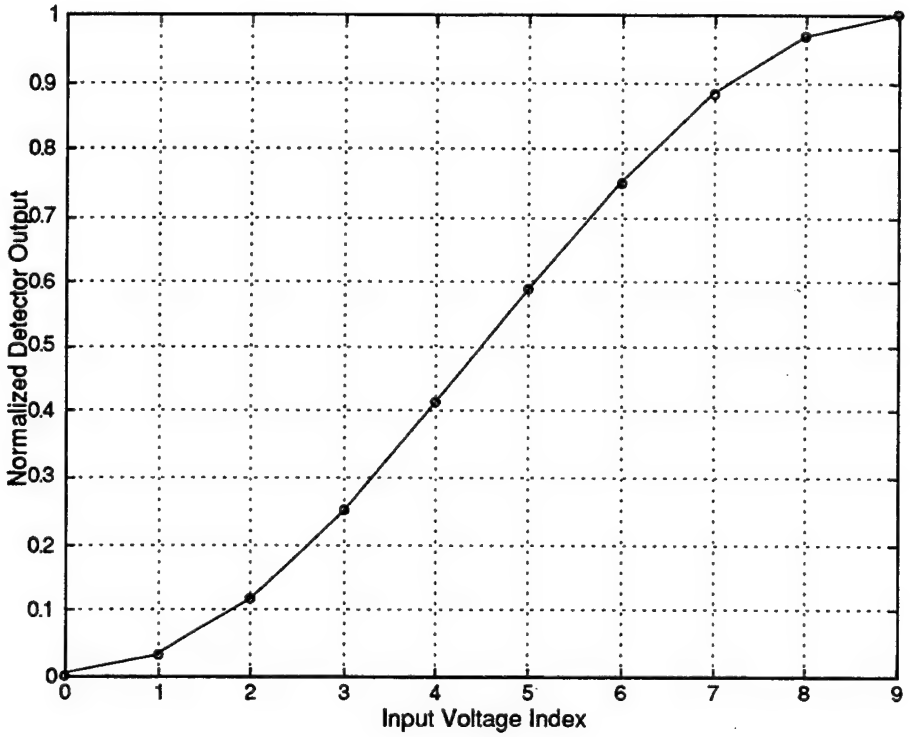


Figure 8. Example of Normalized Threshold Values vs. Input Voltage Index for mod 9

voltages for the various moduli is needed for each system. The Vth_calc.m handles this process also. This program also returns the difference of neighboring normalized threshold values (dVth). Vth and dVth values are required for building up the comparator ladder of the flash ADC scheme for each modulus.

Appendix B shows the program named thmtobin.m which converts the thermometer code generated by the priority encoder into the binary code. This program is needed for confirming operation of the electronic part of the SNS ADC. The user enters the information for each modulus, and the computer returns the possible states of the comparator, thermometer code representation, and the binary information. Using these two programs Vth_calc.m and thmtobin.m, we can get the information needed to construct the SNS ADC circuit.[Ref. 15]

2. Creating the Folding Waveform Inputs Using LabVIEW™

LabVIEW™ is a program development application which uses a graphical programming language. Virtual instruments (VIs) are one of the LabVIEW's powerful features. Using the VIs, we can develop a specific program for data acquisition and instrumental control. LabVIEW is a trademark of National Instruments Corporation. In the early stage of the SNS ADC circuit development, LabVIEW was used to simulate the input signal to the comparators.[Ref. 16,17] For the folding circuits, there are three design rules.[Ref. 11]

Rule (1)

$$M \le 2 \times (m_i)_{min} \times (\text{number of folds}) \tag{12}$$

Rule (2)

$$2^B - 1 \le \prod_i m_i \tag{13}$$

Rule (3)

$$F_{req} \le \frac{v_{max}}{V_\pi} \tag{14}$$

where B is the number of bits, F_{req} is the required number of the folds, v_{max} is the

maximum input voltage, V_π is a half wave voltage for the MZI, and M is the dynamic range of the SNS ADC. Using these relationships, the folding simulation program of LabVIEW is set up.[Ref. 16] The goal of the initial effort is to simulate, and compare the results generated using LabVIEW and the digital hardware to the output of the MATLAB simulation program. Figure 9 shows the folding waveform output of an 8-bit SNS ADC. In this case $m_1 = 9$, $m_2 = 10$, and $m_3 = 11$. This figure was obtained by applying Eq. 3 The x-axis shows the input voltage index and the y-axis shows the normalized value of the MZI output. Note that the initial state of each MZI output is set to the minimum value of the folding waveform.

In LabVIEW, files called JEFFSTEP.VI and CRAIG7.VI are used to simulate the folding circuit output.[Ref. 16,17] LabVIEW creates the step generation of a sine wave with the program files above. In the 8-bit SNS ADC design, we know that the MZI which is used in the actual ADC SNS circuit, has a folding period of $2V_\pi$ or 4.507 V in mod 9, and 32 V is supplied as v_{max}. This maximum voltage gives a sufficient number of folding waves. The number of folds for mod 9 is then calculated using Eq. 14, and the number of folds for the other moduli are calculated by using the ratio of the lowest modulus to the modulus of concern. In this case 9/10 and 9/11 are used respectively for mod 10 and mod 11. Let F_m be the number of folds for mod m. Then F_9, F_{10}, and F_{11} are calculated from Eq. 8 as follows:

$$F_9 \le \frac{2(32)}{4.507} = 14.20 \tag{15}$$

$$F_{10} = F_9 \times \frac{9}{10} = 12.78 \tag{16}$$

$$F_{11} = F_9 \times \frac{9}{11} = 11.62 \tag{17}$$

Using the cycle ratio for each folded output waveform, the frequency ratios are calculated. The files, JEFFSTEP.VI and CRAIG7.VI use 110, 99, and 90 cycles for mod 9, 10 and 11 respectively in the 8-bit design. The number of points of observation (#obs) is determined by following[Ref. 16]:

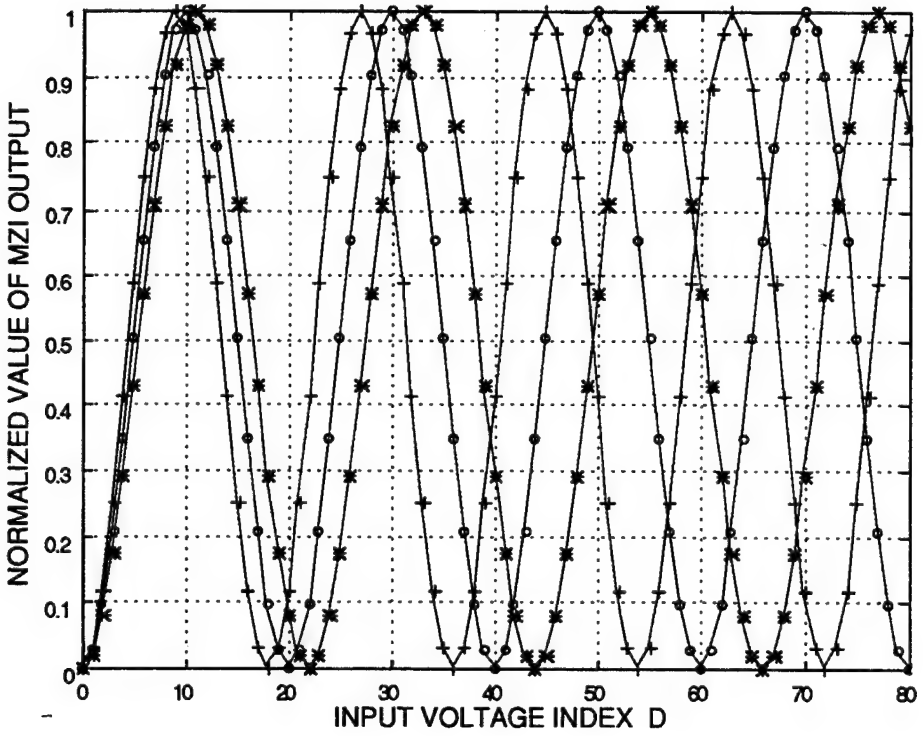


Figure 9. The Folding Waveform Output of an 8-bit SNS ADC

$$(\#\text{obs}) = \frac{\text{Number of generation points}}{\text{lowest modulus cycles}} \times F_{\text{lowest modulus}} \tag{18}$$

5,000 is initially chosen as the number of generation points on both JEFFSTEP.VI and CRAIG7.VI to represent the whole dynamic range. Since the number of cycles associated with the lowest modulus (mod 9) is 110, 5,000/110 corresponds to one fold of mod 9. Therefore, the number of points of observation for the 8-bit design, which requires only 256 states, is 646 by Eq. 15, and Eq. 18 as follows:

$$\begin{aligned} \frac{\text{Number of generation points}}{\text{lowest modulus cycles}} \times F_9 &= \frac{5{,}000}{110} \times 14.20 \\ &= 645.5 \end{aligned} \tag{19}$$

The step input increment (size of the least significant bit) V_{LSB} is determined to be 0.250 V by Eq. 7. The size of the least significant bit for each modulus is the same. In order to compare with the LabVIEW results, we call the number of observation points per input voltage index the number of sample points. The number of sample points ($\#_{\text{sample}}$) must be calculated and should also be the same for each modulus, since each modulus has same sample points in a system. This number of sample points can be computed from

$$(\#_{\text{sample}}) = \frac{\text{number of generation points}}{\text{number of cycle} \times 2 \times m} \tag{20}$$

For example using 5,000 points, 110 cycles, and m = 9, Eq. 20 provides $\#_{\text{sample}} = 2.5$.

The MATLAB code is programmed using the information above. Appendix C shows the simulating LabVIEW program file. This MATLAB file named Lab_simu.m uses the following input information.

- Number of generating points
- Number of points of observation
- Number of cycles of each modulus
- Number of each modulus

This information provides the folding circuit information as

- The number of folds of mod m, F_m
- The number of sample points between two thresholds of each comparator
- The thermometer code for each sample point of each modulus
- Predicted normalized LabVIEW output at each sample point

Input information for this program should be prepared in advance. For example, an 8-bit SNS ADC design using mod 9, 10, and 11 with 5,000 generating points, the input information is as follows:

- Number of generating points : 5,000
- Number of points of observation : 647 (from 0 to 646)
- Number of cycles of each modulus : 110, 99, 90
- Number of each modulus : 9, 10, 11

3. Simulation Results

After programming the file Lab_simu.m, the output of both the MATLAB simulation and LabVIEW are compared. The LabVIEW program named CRAIG7.VI creates the output value, and either displays or prints out the numbers. This file uses 5.0 V as a maximum voltage of the folding circuit, so each voltage value of the index obtained by the MATLAB simulation must be multiplied by five. The output of the MATLAB simulation always provides a normalized value. CRAIG7.VI calculates three decimal places instead of the four decimal places calculated with the MATLAB simulation. Table 2 shows the comparison of some of the output values generated with CRAIG7.VI and Lab_simu.m.

As a result, the MATLAB simulation can simulate the LabVIEW generated signals. This program can also simulate the thermometer code output for each modulus. These outputs are very helpful to confirm the output states of each modulator. On an actual 8-bit SNS ADC board, three LED displays are used for confirming these output states. The MATLAB simulation can also be used to create an arbitrary configuration of an SNS ADC design. The file, Lab_simu.m, is based on the interferometer combination of mod 9, 10, and 11 with 5,000 generating points, and we can apply this file to a new combination simply by changing the input information.

Index	LabView (mod 9)	LabView (mod 10)	LabView (mod 11)	MATLAB simulaion (mod 9)	MATLAB simulaion (mod 10)	MATLAB simulaion (mod 11)
1	0	0	0	0	0	0
2	0.024	0.019	0.016	0.0238	0.0193	0.0160
3	0.095	0.077	0.064	0.0949	0.0770	0.0637
4	0.212	0.172	0.143	0.2119	0.1721	0.1425
5	0.373	0.303	0.251	0.3725	0.3032	0.2515
6	0.574	0.468	0.389	0.5737	0.4683	0.3892
7	0.812	0.665	0.554	0.8117	0.6647	0.5538
8	1.082	0.890	0.743	1.0818	0.8896	0.7434
9	1.379	1.139	0.955	1.3790	1.1393	0.9554
10	1.698	1.410	1.187	1.6976	1.4100	1.1871
11	2.032	1.698	1.436	2.0315	1.6976	1.4356
12	2.374	1.998	1.698	2.3744	1.9976	1.6976
13	2.720	2.305	1.970	2.7196	2.3054	1.9700
14	3.061	2.616	2.249	3.0607	2.6162	2.2491
15	3.391	2.925	2.531	3.3910	2.9252	2.5314
16	3.704	3.228	2.813	3.7044	3.2276	2.8133
17	3.995	3.519	3.091	3.9948	3.5188	3.0912
18	4.257	3.794	3.362	4.2566	3.7942	3.3616
19	4.485	4.050	3.621	4.4850	4.0496	3.6210
20	4.675	4.281	3.866	4.6755	4.2810	3.8660
21	4.824	4.485	4.094	4.8244	4.4850	4.0936
22	4.929	4.658	4.301	4.9291	4.6582	4.3008
23	4.987	4.798	4.485	4.9874	4.7981	4.4850
24	4.998	4.902	4.644	4.9982	4.9025	4.6438
25	4.961	4.970	4.775	4.9614	4.9697	4.7753
26	4.878	4.999	4.878	4.8776	4.9988	4.8776

Table 2. Comparison of Program Output Value with LabVIEW and MATLAB

C. EXTENDED DEVELOPMENT OF MATLAB PROGRAM

1. Overview of the Parity Circuit

The SNS ADC simulation program, developed in the previous section, is based on the original SNS ADC design shown in Fig. 6 in Chapter I. This original design, however, results in a number of encoding errors.

In the original design of the SNS ADC, it is assumed that the state of each modulus changes simultaneously. Figure 10 shows details of a folding waveform of an 8-bit SNS ADC. The important thing about the SNS ADC is that the folding waveforms for each modulus pass through the threshold values simultaneously across all channels. This is demonstrated by the vertical line drawn at the input voltage index of six. Simultaneous transition of all modes from one state to another is the ideal situation. However, this may not happen in an actual circuit because of the inaccuracies of the matching voltages. When the folding waveforms do not pass through their matching threshold levels simultaneously, an encoding error results.

To eliminate the possibility of these encoding errors, a parity circuit is used to set up a band around each code transition point. This is illustrated in Fig. 11 for one threshold level. The circuit modification requires a few additional comparators in the smallest channel. That is, $2m_1$ comparators are used instead of m_1 -1. Of these, $2m_1$ - 2 would be paired, one just below the proper comparator threshold level and the other just above. Of the remaining two, one would be just above the minimum modulation depth, and the other threshold just below the maximum detector output. A schematic diagram of the SNS ADC including the parity circuit is shown in Fig. 12. The operation of this parity circuit is as follows:

Assume that a signal arrives and all comparators below a certain level are on. This is as it should be. For each incoming signal, if it turns on an *even* number of comparators, 0,2,4, ..., m_1, the signal is rejected. If it turns on an *odd* number of comparators, it is accepted. In this manner we place a narrow band around each voltage level that corresponds to a switching point. For all moduli, the comparator levels are adjusted so that they cross their respective folding waveforms within these narrow bands. Any input signal within these narrow rejection bands can be easily discarded. The parity circuit operation

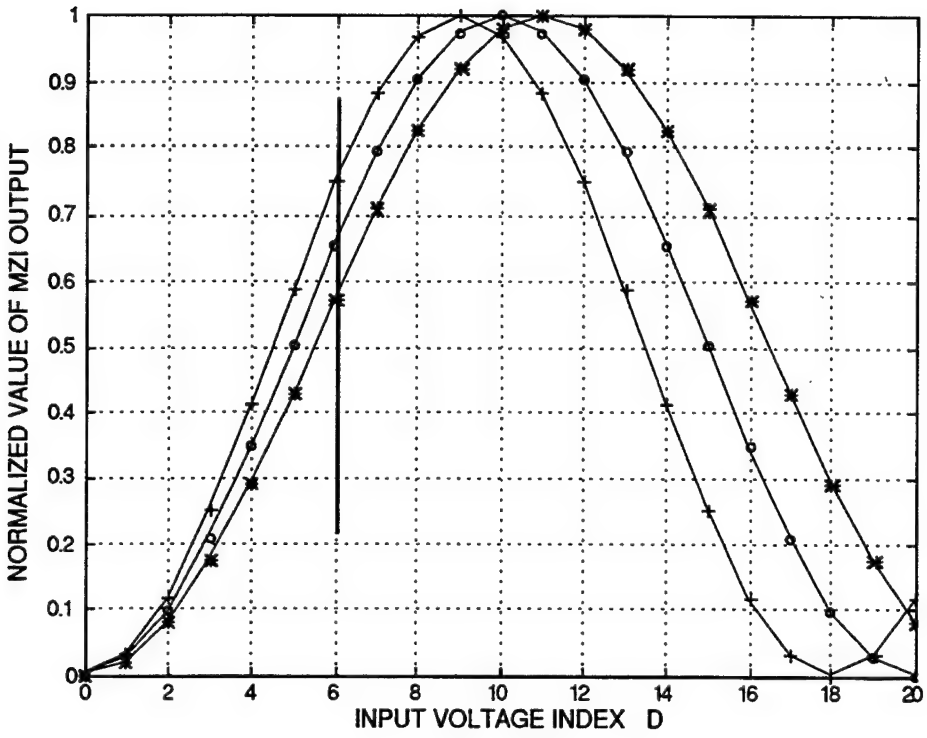


Figure 10. Details of the Folding Waveform of 8-bit SNS ADC

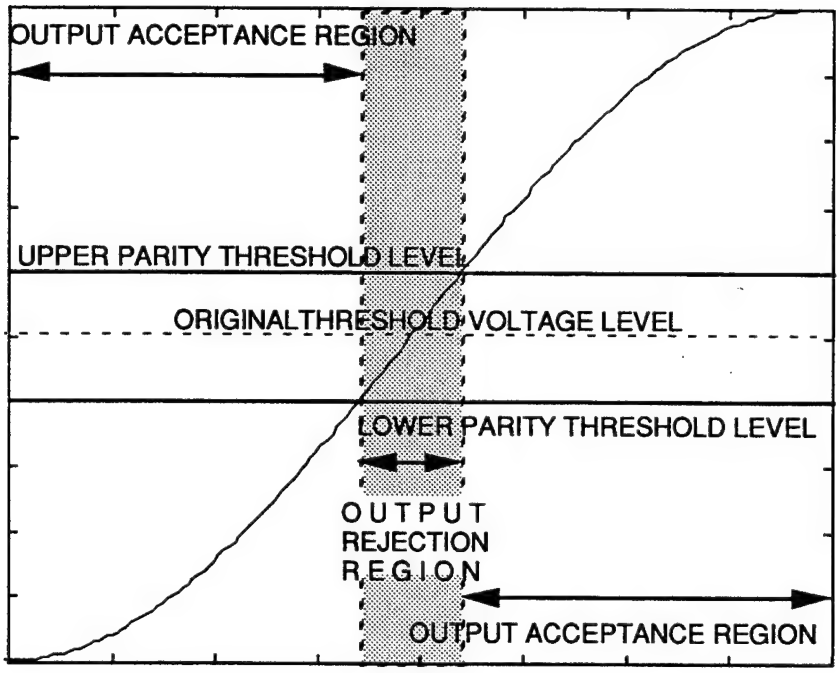


Figure 11. The Concept of Parity Circuit

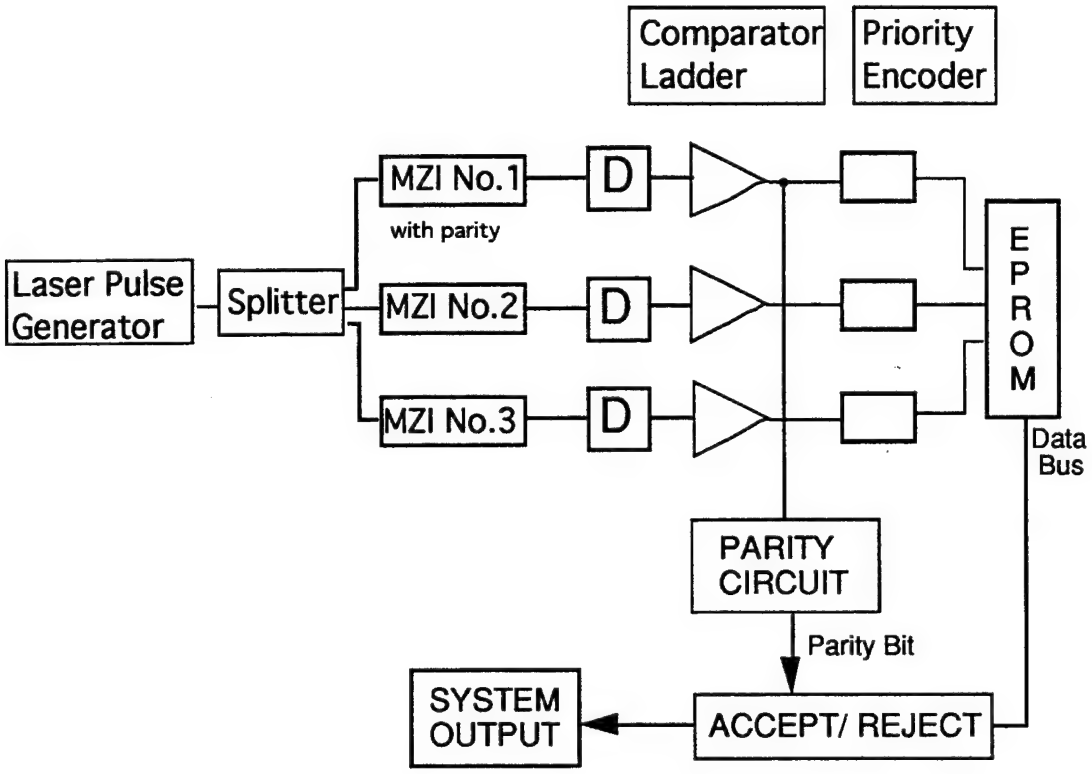


Figure 12. Block Diagram of the SNS ADC Including the Parity Circuit

region of mod 9 is shown in Fig. 13. The gray shaded region indicates the rejection area. Samples in this region will be ignored. The unshaded areas indicate the acceptance region. Legitimate state combinations will be accepted as a valid output. The dotted horizontal line is the original threshold level of mod 9. The solid line above and under the dotted line shows the upper and lower threshold levels for the parity comparators.

2. Parity Circuit Threshold Voltages

To set up the parity circuit threshold levels, we must select reasonable rejection (decimation) regions. First we need to get V_{LSB}. Once V_{LSB} is obtained, the width of the decimation region can be calculated from the user's width specified as a percentage of the LSB width.

$$(\text{decimation width}) = \frac{(\%\text{desired})}{100} \cdot V_{LSB} \tag{21}$$

The upper and lower parity circuit input voltage for input voltage index k are chosen as follows;

$$(\text{upper parity})_{in,\,k} = V_{in,\,k} + \frac{1}{2}(\text{decimation width}) \tag{22}$$

$$(\text{lower parity})_{in,\,k} = V_{in,\,k} - \frac{1}{2}(\text{decimation width}) \tag{23}$$

where $V_{in,k}$ is an input voltage corresponding to the input voltage index k. Finally we can compute the 2(m-1) +2 parity circuit normalized threshold levels as

$$(\text{upper parity})_{out} = \sin^2(\text{upper parity})_{in} \tag{24}$$

$$(\text{lower parity})_{out} = \sin^2(\text{lower parity})_{in}. \tag{25}$$

Figure 14 shows an example of this parity circuit computation. In this example, mod 3 is

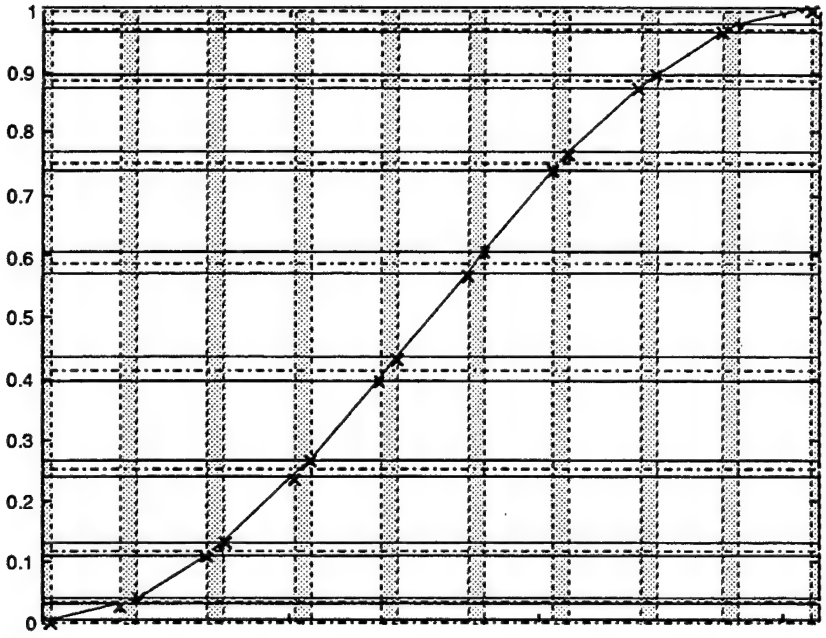


Figure 13. Parity Circuit Operational Region in mod 9

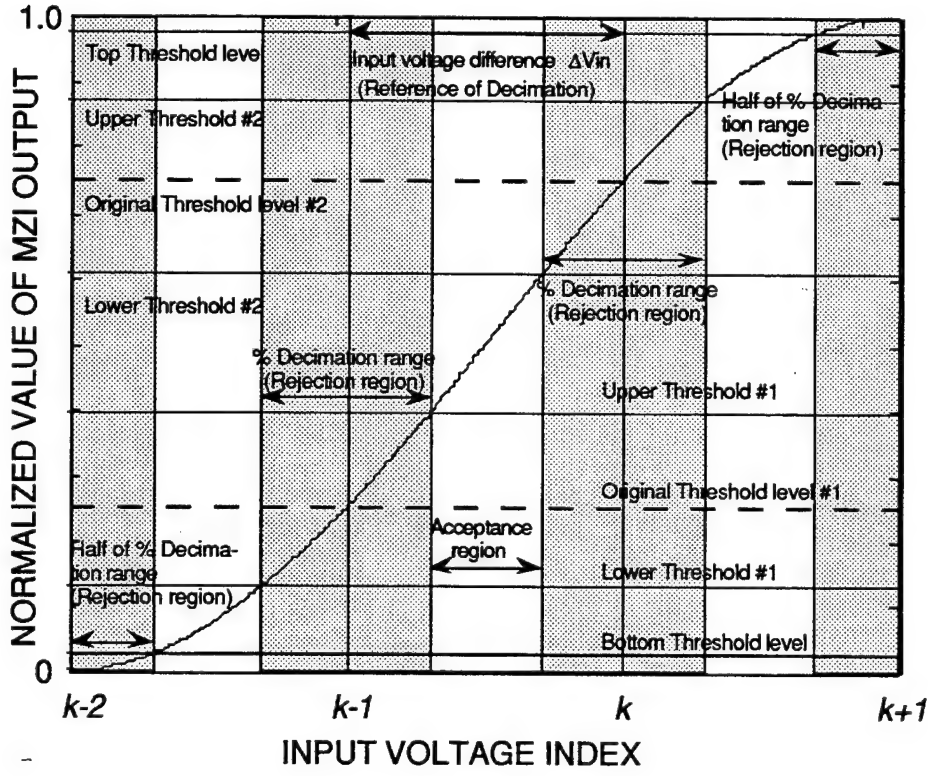


Figure 14. Example of the Parity Circuit Threshold Computation (mod 3)

used for the smallest channel, and the decimation width is 60% of V_{LSB}. The larger the percentage of decimation width chosen, the wider the threshold levels between the parity and rejection range.

The MATLAB program, idlprty.m, covers these processes. Idlprty.m is given in Appendix Section D. The process to set up the parity circuit threshold levels is similar to the program Vth_calc.m. Desired percentage of the rejection region and the modulus number of the least significant bit are needed. Using the decimation width and the modulus corresponding to the smallest channel, the program first computes the input voltage indices and V_{LSB}, and the original threshold values. The threshold levels for the parity circuit are calculated in the next stage, and finally, the output values of the parity circuit are computed. Figure 15 shows the threshold levels for a smallest channel of mod 9 with a 20% decimation width.

3. Simulating the System with the Parity Circuit

The final goal of the MATLAB SNS ADC simulation is to simulate the complete system using the LabVIEW signals. Required output includes:

1. Threshold level for each modulus
2. The input voltage samples generated by LabVIEW
3. Output state related to each input voltage sample
4. Parity circuit threshold levels of the smallest modulus
5. The closest states and the difference for each sample output
6. Parity bit information (accept / reject)

The first four items are completed by Vth_calc.m, Lab_simu.m, and idlprty.m. The MATLAB file, sys_simu.m is shown in Appendix Section E and runs all three of these files. In order to meet the fifth requirement above, the following technique is used. Initially, two vectors are created. The vector named AdiffmVout shows the difference between the output of each sample index and the closest threshold just below the sample for a positive slope of the folding waveform, or the difference between the output of each sample index and the closest threshold just above the sample for a negative slope of the folding waveform of mod m. BdiffmVout shows the difference between each sample index and its closest threshold just above the sample for a positive slope folding waveform, or the difference between each sample index and its closest threshold just below the sample for a negative slope folding waveform of mod m. Once obtained, these two vectors are

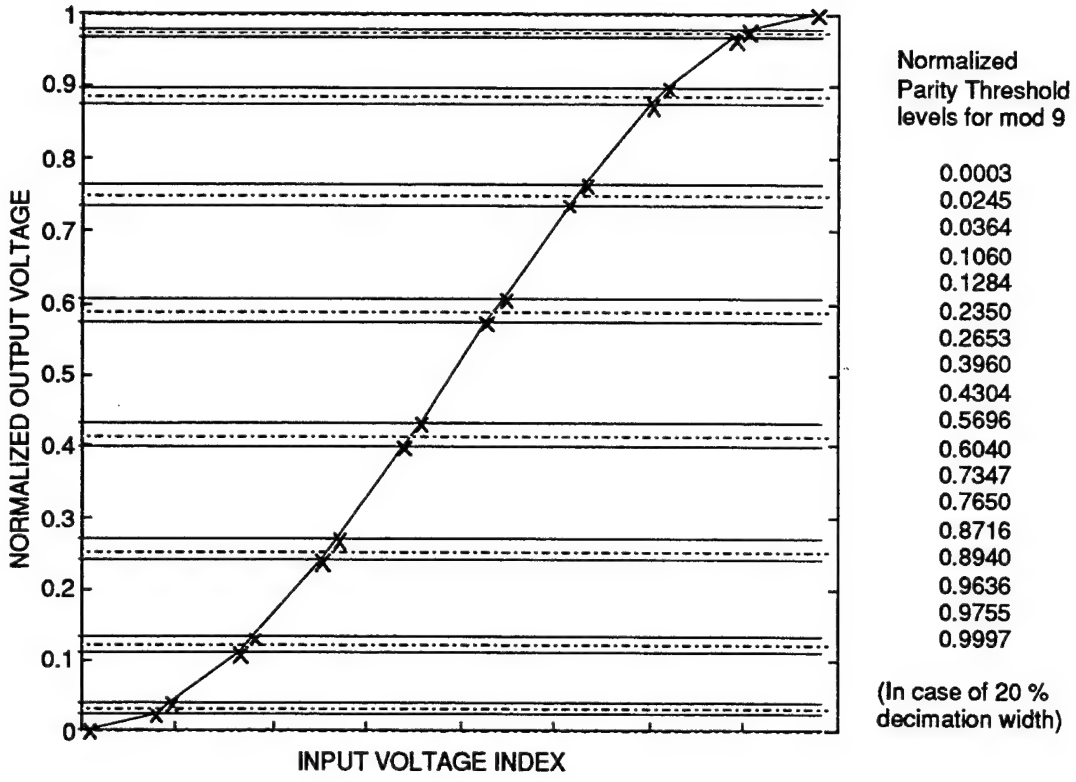


Figure 15. Example of the Parity Circuit Threshold Computation (mod 9)

compared with each other. The smallest value is chosen, and the new vector, diffmVout is created for mod *m*. Also, the vector named closeVth*m* shows the closest threshold level for each sample point of mod *m*. To satisfy the last requirement, the decimation region is initially set up for the smallest channel (mod 9 is chosen as the smallest channel in our project design). The vector diff9Vout contains information on the difference between the sample index and the closest threshold value. For the upper parity threshold level, we reject the sample if that difference is smaller than the difference between the original threshold level and the upper parity threshold level. For the lower parity threshold level, we reject the sample if that difference is smaller than the difference between the original threshold level and the lower parity threshold level. In all other cases, we accept the sample output. If the sample is in the rejection region, the parity bit shows a "1" (even parity), and if the sample is in the acceptance region, the parity bit shows a "0" (odd parity). Finally, this information is stored in a file named 'diary'. Using the file, sys_simu.m, we can simulate the complete 8-bit SNS ADC system, and we can compare the actual hardware performance using the LabVIEW generated input signals. This testing and comparison is described in Chapter IV.

III. PARITY TIMING PROCESSOR

A. OVERVIEW

In the SNS ADC hardware, there is a considerable difference in the arrival time of the parity bit and the digital word representing the analog input. To engineer the path length to be the same, a parity timing processor is constructed.

B. DEVELOPMENT OF THE TIMING PROCESSOR

Figure 16 shows the system diagram of the SNS ADC that includes the parity circuit information path, as well as the timing processor diagram. The timing processor is divided into two parts: the clock circuit and the holding circuit. The timing processor uses the same detected laser pulse as an input signal. The AND gate creates the clock signal for the D-Q latching circuit. The parity circuit signal is designed such that an even parity (bit information of "0") gives an "ON", and odd parity (bit information of "1") gives an "OFF". So when both the clock pulse and parity circuit signal are "ON", the clock signal is generated and drives the D-Q latch. Otherwise, the previous output is held until the next clock pulse. Figure 17 shows the schematic diagram of the timing processor. The timing processor operates as follows: The regulator chip LM7805CT generates the stable +5 Vdc from +12 Vdc to supply Vcc for all IC chips. The comparator AD 9698 compares the pulse input from the photo detector with the threshold level voltage, and gives a TTL level pulse output. The LM324 is used as a buffer for the threshold voltage, which is generated by a 10 kΩ potentiometer and a 100 Ω resistance. The PWC-30 accepts the TTL level pulse and generates an output pulse of set duration. 74S08 is an AND logic gate which generates the clock pulse to drive the D-flip flop 74ALS374. The 74ALS374 latches the digital word from the EPROM. The 74ALS374 is the final stage of the SNS ADC. The following subsection describes each device's role in the timing processor and specifications.

1. Laser Transmitter, *Model 400*

The laser transmitter should have as much power as practical to supply the system with sufficient transmit power. From the previous research done by LT. Craig Crowe, the model 400, manufactured by Broadband Communication Products, Inc., is chosen as a

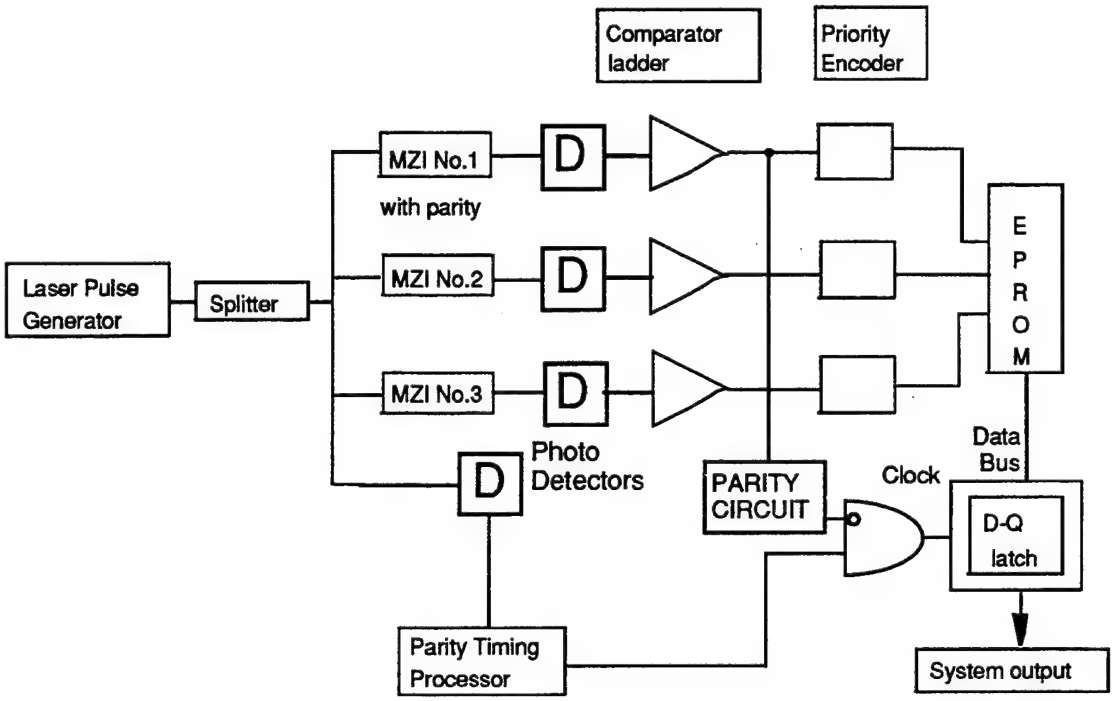


Figure 16. Modified Block Diagram of the SNS ADC

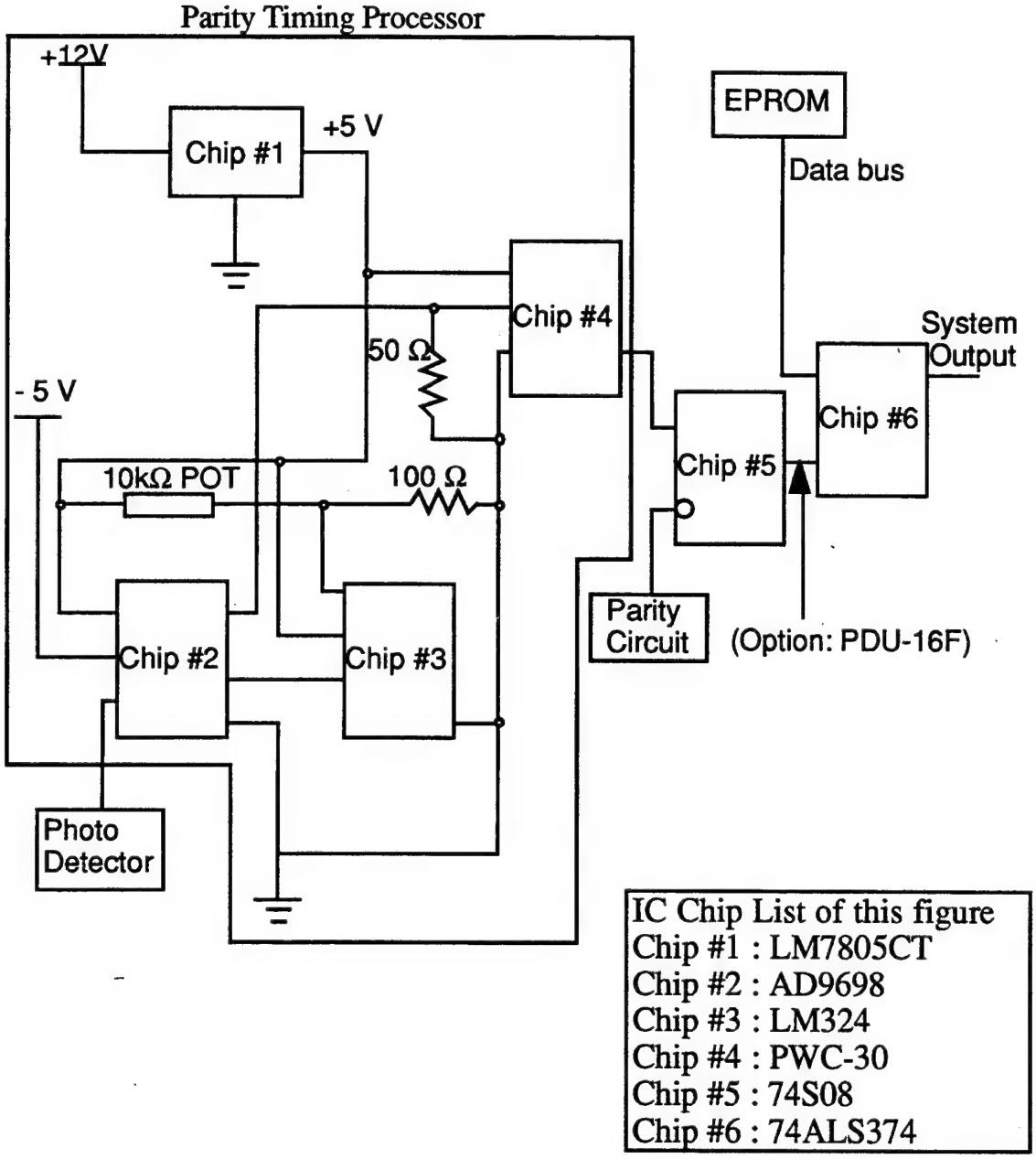


Figure 17. Schematic Diagram of the Parity Timing Processor

laser transmitter for this design.[Ref. 17] The model 400 is a high speed optical signal source with a data rate as high as 1.3 giga bit per second (GBPS). The model 400 can also handle both digital and analog inputs up to 1 GHz. In the digital mode, the model 400 accepts an input pulse pattern from DC to sub-nanosecond pulses in continuous or burst mode, and the pulse width of the optical output and laser bias point can be adjusted. Table 3 shows the characteristic table for the model 400.[Ref. 18] The model 400 provides the specified data rate and pulse width required for this application. LT. Crowe also mentioned that the mode lock laser would be a better choice for the laser transmitter because it offers a significant improvement in power over that of the model 400. The laboratory's capability, however, requires us to use the model 400.[Ref. 18] The optical pulse generated by the model 400 goes to a 1 x 4 splitter. Three of the pulses go to the folding circuits, and the remaining one is detected and goes to the timing processor.

2. Photo Detector, *Model 1811*

The photo detector is an optical to electrical converter, or optical receiver. For each folding circuit modulator, the wideband photo detector model 310A was selected because it had the higher gain and bandwidth of any of the equipment Lt. Crowe investigated. This receiver includes a variable gain. These features are important for the folding circuits, but are not required for the timing processor design. A certain level of gain is required in this case (but it does not have to be same gain as model 310A): We also must consider the limitation of the laboratory's capability. For this reason, model 1811, manufactured by New Focus, Inc., is chosen. The model 1811 is a low-noise photo detector which uses an InGaAs PIN photo diode in the near IR wavelength region (800 nm to 1800 nm). This is a range which is includes the wavelength of the laser transmitter. In addition, the model 1811 is more compact than the model 310A. The specifications of model 1811 are shown in Table 4.[Ref. 19] The photo detector detects the optical signal and converts it to an electrical signal which passes to the timing processor.

3. Fast Speed Comparator, *AD9698*

When constructing an analog-to-digital converter, the performance of the comparator plays the key role in the total system. The AD9698, manufactured by Analog Devices, Inc., is designed as an ultra fast TTL-compatible voltage comparator. This chip can achieve propagation delays previously possible only in high performance emitter-coupled logic (ECL) devices. ECL's offer faster performance than TTL, but have

Parameter	Characteristic
Digital Input (Electrical)	
Bit Rate	1.3 GBPS
Pulse Pattern	No Constraint, any combination
Digital Output (Optical)	
Pulse Rise/Fall time	0.5 nsec (max.)
Minimum Pulse Width	0.75 nsec
Extinction Ratio	10 : 1 (min.)
Peak Coupled Power	0.75 mW into (8/125) fiber
Wavelength	1300 nm
Spectral Width	4 nm
Analog input(Electrical)	
Frequency	0.1-1,000 MHz
Input level	1.0 V peak to peak (typ.)
Input impedance	50Ω, AC coupled
Analog Output (Optical)	
Average Optical Power	0.5 mW
Frequency Response	0.1-1,000 MHz
Polarity	Inverting

Table 3. Specifications of the BCP model 400

Parameter	Characteristic
Coupling	DC or AC
Bandwidth (3 dB)	DC-125 MHz (typ. DC) 25 kHz-125 MHz (typ. AC)
Wavelength Range	800-1800 nm
Photodiode Material	InGaAs PIN
Power Requirements	±15 Vdc; 250 mA
Rise Time	3 nsec (typ.)
Current Gain	40 V/mA (typ.)
Input Noise Current	2 pA/Hz(1/2) @ 10 MHz
Output Current	40 mA (max into 50 Ω)
Maximum Input Power	5 mW (max @ 1.3 μm)

Table 4. Specifications of the model 1811 (New Focus, Inc.)

problems with the transmission-line effect due to the length and inductance of the signal lines. Table 5 shows the switching performance of the AD9698. Rise time, t_s, is one of popular specifications used to show the response speed of the comparator. Previous TTL-compatible fast comparators such as LT1016 (manufactured by Linear Technology Corp.) or Am686 (manufactured by Advanced Micro Devices) have 6 to 10 nsec of t_s, while AD9698 has 1.85 nsec (typ.). [Ref. 1: pp. 584-585, Ref. 20] Table 6 shows the comparison of the TTL-compatible comparator with other comparators. In the high speed 8-bit SNS ADC design, many AD9698 comparators are used in the comparator ladder circuit. In the timing processor, the AD9698 is used as the threshold detector. The operation of the pulse detector is shown in Fig. 17. The transmitted laser pulse generated by the BCP model 400 comes to the Photo Detector, New Focus model 1811, which converts the optical signal to an electrical pulse. The amplitude of this pulse is small compared to the TTL driving level. The threshold value of the comparator is set as a small value (several tens of mV). The comparator compares these small values and creates an output signal at the TTL compatible level. For setting the threshold level, a 10 turn -10 kΩ potentiometer and a 100 Ω resistor are used to vary the comparator's threshold level in the mV region. The comparator is the front end of the timing processor with its output going to the pulse width controller PWC-30.

4. Pulse Width Controller, *PWC-30*

The TTL level comparator output is sent to the pulse width controller, PWC-30 series, manufactured by Data Delay Device, Inc. During the process of building up the timing processor, it was found that the AD9698 has a TTL level output representative of a spike. This waveform causes the timing processor to mistrigger. The short pulse width of the signal does not have enough time to hold the TTL level output "HIGH" till the following stage starts operating. If the pulse cannot hold its output level on HIGH for at least the rise time of the next device, the system will have no response. In order to avoid these problems, a pulse width controller is needed. Table 7 shows the specifications of the selected PWC-30 series. The features of the PWC-30 are as follows:

- Exact control of pulse width (min. 5 nsec to max. 500 nsec)
- Rising edge trigger
- Low power consumption
- Auto-insertable

Parameter	Testing temperature	Characteristics
Propagation Delay		
Input to Output HIGH Input to Output LOW	0 to 70 °C	4.5 nsec (typ.)
Latch Enable Input to Output HIGH Latch Enable Input to Output LOW	0 to +70 °C	6.5 nsec (typ.)
Delta Delay Between Outputs	+25 °C	0.5 nsec (typ.)
Propagation Delay Dispersion		
20 mV to 100 mV Overdrive 100 mV to 1.0 V Overdrive	+25 °C	0.1 nsec (typ.)
Rise Time	+25 °C	1.85 nsec (typ.)
Fall Time	+25 °C	1.35 nsec (typ.)
Latch Enable		
Pulse Width	+25 °C	2.5 nsec (typ.)
Setup Time	+25 °C	1.7 nsec (typ.)
Hold Time	+25 °C	1.9 nsec (typ.)

Table 5. Switching Performance of AD9698 (Analog Devices, Inc.)

Chip type	Am686	LT1016	EL2019C	AD9698
Quantity per package	1	1	1	2
rise-time (typ.)	9 (nsec)	10 (nsec)	6 (nsec)	1.85 (nsec)
Does the chip cover Q and Q' output ?	no	yes	no	yes
Is the chip O.K. for supplying single 5 V ?	no	yes	no	yes

Table 6. Comparison of the Fast TTL Level Comparator

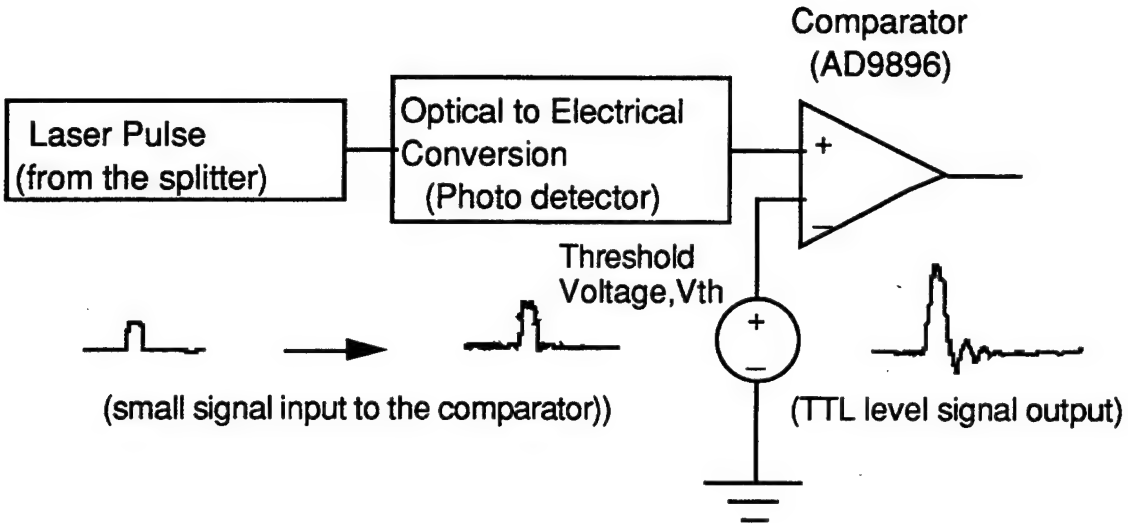


Figure 18. Operation of the Threshold Detector

Parameters	Characteristics
Trigger inherent delay	$T_{DO} = 6.5 \pm 1$ nsec
Pulse-width tolerance	5% or 2 nsec
Maximum Input PRR	PW + 20 nsec
Output rise-time	2 nsec (Typ.)
HIGH-level input voltage	2.0 V (min.)
LOW-level input voltage	+ 0.8 V (max.)
HIGH-level output voltage	3.4 V (typ.), 2.7 V (min.)
LOW-level output voltage	+0.5 V (max)

Table 7. Specifications of the PWC-30 (Data Delay Device, Inc.)

- High speed

Because it uses a rising edge trigger, the PWC-30 can avoid the multiple trigger problem. Figure 19 shows the block diagram of the PWC-30. In this timing processor, the PWC-30-50, generates 50 nsec of pulse width.

5. Programmable Delay Units, *PDU-16F*

This device is an optional device in the timing processor. At the final stage of the SNS ADC, the D-Q latch should handle both the digital word information from the EPROM and the clock signal. The data path which handles the digital word and the path of the timing processor signal do not have same propagation delay. The EPROM has the largest propagation delay (around 100 nsec) in the data path, while the timing processor has much less of a propagation delay. In order to get both data and timing information at the same time, we need to insert delay devices. There are many options to set up a delay time. An easy way is to use an optical delay line (ODL). In this method, an additional length of fiber cable is inserted in the optical signal line to get the desired time delay. The ODL, however, offers little flexibility in tuning the delay time. One ODL corresponds only to one fixed delay time. During the process of building up the timing processor, we cannot expect to know the exact time delay of both the data and timing signal. Using an electrical delay unit is another way of creating a specific delay time. Data Delay Devices, Inc., offers various types of delay units. Programmable delay units (PDU) were chosen as the electrical delay line device. The PDU series is a fast logic programmable delay unit. The PDU has a 3-bit to 8-bit programmable delay line, and we can choose the suitable chip among the various incremental delay step sizes. In this timing processor, PDU-16F, which has 6-bit programmable delay lines, is used. The block diagram of PDU-16F is shown in Fig. 20. For the timing processor, we prepare PDU-16F-2, which has 2 ± 0.5 nsec of incremental delay for each step and a total programmed delay of 126 nsec. The specifications of the PDU are shown in Table 8.

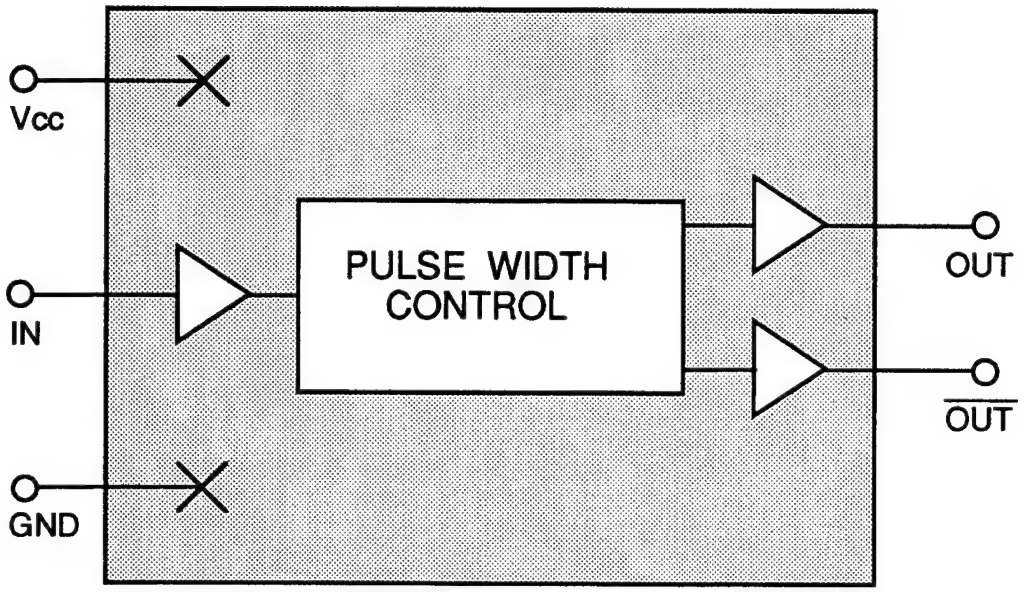


Figure 19. Block Diagram of the PWC-30 (Data Delay Devices, Inc.)

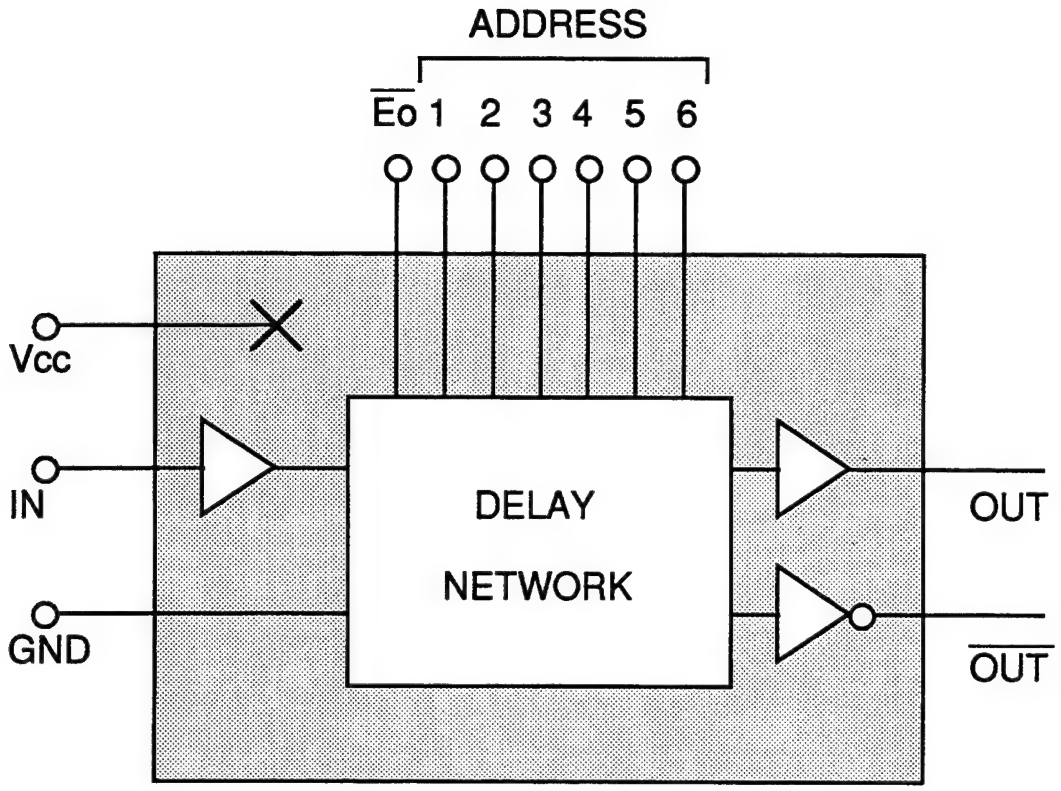


Figure 20. Block Diagram of the PDU-16F (Data Delay Devices, Inc.)

Parameter	Characteristics
Propagation Delay	
Address to output (T_{SUA})	7 nsec (typ.)
Enable to output (T_{SUE})	6 nsec (typ.)
Delay variation	Monotonic in one direction
Programmed delay tolerance	± 5% or 1 nsec
Inherent delay	9 nsec (pin number 2) (typ.) 8 nsec (pin number 1) (typ.)
Minimum pulse-width	10% of total delay
HIGH-level output voltage	2.5 V(min.) 3.4 V (typ.)
LOW-level output voltage	0.5 V(max.) 0.35 V(typ.)
HIGH-level input voltage	2.0 V (min.)
LOW-level input voltage	0.8 V (max.)

Table 8. Specifications of the PDU-16F (Data Delay Devices, Inc.)

IV. RESULTS

A. 8-BIT SNS ADC PERFORMANCE

1. Overview of Decimation

The original SNS ADC can present a problem at each of the code transition points. The probability of this occurrence depends upon the offset voltage match of the comparators, the tolerance of the voltage dividing resistors, and the corresponding width of the input voltage region being affected. The modified SNS ADC with a parity circuit, is introduced to solve this problem. The parity circuit is used to determine if the sample lies at a code transition point. The goal of this section is to test the modified SNS ADC hardware using the LabVIEW-generated input. Also, a comparison of the actual circuit response to the simulation is investigated. Several decimation widths are tested to determine the effectiveness of the parity circuit.

2. Testing Configuration

The system configuration for testing the circuit is shown in Fig. 21. We use an 8-bit SNS ADC with the LM311 general purpose comparator, which has a 200 nsec (typ.) response time. The SNS ADC uses a "static" input sampling signal generated by the LabVIEW program, CRAIG7.VI. For a "static" signal, each input voltage level has a certain time period which is relatively long. For CRAIG7.VI, we choose a time step of 0.5 sec with the LabVIEW signal output changing levels at each new time step. The system moduli for the SNS ADC are mod 9, 10, and 11. The corresponding total number of comparators is 37 with a maximum of 18 loaded in parallel. The mod 9 channel is used for the parity circuit, and the chosen decimation widths of 0% (no decimation), 10%, 15%, 20%, and 30% were investigated using the corresponding resistance networks. Values for the folding circuit resistance ladder (except for the parity circuit), and the target threshold voltages, are shown in Table 9. The values for the parity circuit resistance ladder and target threshold voltages are shown in Table 10. No timing circuit was required for these tests, since the LabVIEW program controls the input voltage sequence, and this is a static analysis of the system. The signal flow of the testing is as follows: First, CRAIG7.VI creates the input folding waveforms from the SNS ADC with a 0.5 sec time step between

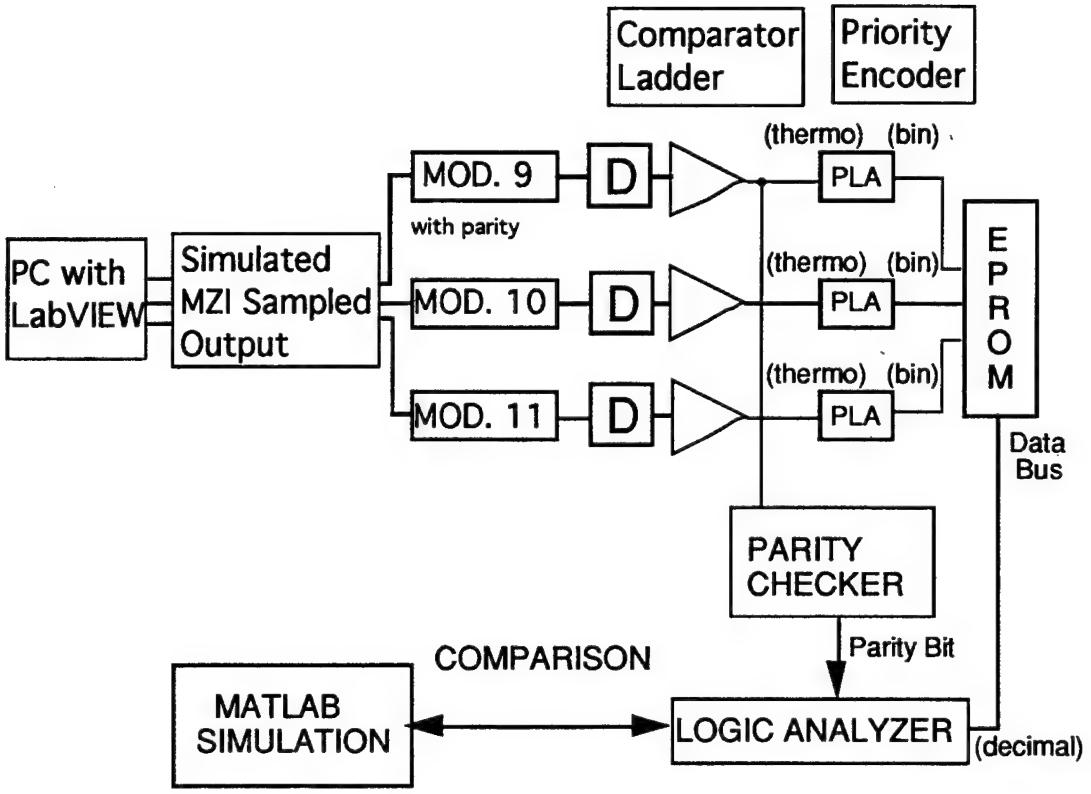


Figure 21. Block Diagram of Testing 8-bit SNS ADC Circuit

	mod 10	mod 11
Reference Voltage	5.00 (V)	5.00 (V)
Signal Input Range	0 to 5.0 (V)	0 to 5.0 (V)
Ladder resistance (kΩ)	**total resistance R_{tot} = 100 kΩ for modulus**	
Resistance No.1	2.45	2.03
Resistance No.2	7.10	5.91
Resistance No.3	11.10	9.32
Resistance No.4	11.90	11.97
Resistance No.5	15.50	13.66
Resistance No.6	15.50	14.23
Resistance No.7	11.90	13.66
Resistance No.8	11.10	11.97
Resistance No.9	7.10	9.32
Resistance No.10	2.45	5.91
Resistance No.11	--	2.03

Table 9. Testing Configuration of Folding Circuit (Value of Resistance Ladder)

	mod 10	mod 11
Target Threshold (V)		
Threshold Voltage No.1	0.123	0.102
Threshold Voltage No.2	0.478	0.397
Threshold Voltage No.3	1.031	0.863
Threshold Voltage No.4	1.728	1.462
Threshold Voltage No.5	2.500	2.144
Threshold Voltage No.6	3.273	2.856
Threshold Voltage No.7	3.970	3.538
Threshold Voltage No.8	4.523	4.137
Threshold Voltage No.9	4.878	4.603
Threshold Voltage No.10	--	4.899

Table 9.(cont.) Testing Configuration of Folding Circuit (Threshold Voltage)

mod 9 (Parity circuit)	0 % decimation	10 % decimation	15 % decimation	20 % decimation	30 % decimation
Resistor	R_{tot} = 100 kΩ for each case				
R1	--	10	17	30	68
R2	3,020	2,716	2,567	2,417	2,116
R3	--	597	895	1,194	1,790
R4	8,680	7,822	7,390	6,958	6,094
R5	--	1,122	1,683	2,244	3,364
R6	13,300	11,983	11,323	10,661	9,336
R7	--	1,511	2,267	3,022	4,532
R8	16,320	14,700	13,890	13,078	11,452
R9	--	1,719	2,578	3,437	5,154
R10	17,360	15,643	13,890	13,078	11,452
R11	--	1,719	2,578	3,437	5,154
R12	16,320	14,700	13,890	13,078	11,452
R13	--	1,511	2,267	3,022	4,532
R14	13,300	11,983	11,323	10,661	9,336
R15	--	1,122	1,683	2,244	3,364
R16	8,680	7,822	7,390	6958	6,094
R17	--	597	895	1,194	1,790
R18	3,020	2,716	2,567	2,417	2,116
R19	--	10	17	30	68

Table 10. Testing Configuration of Parity Circuit (Value of Resistance Ladder)

mod 9 (Parity circuit)	0 % decimation	10% decimation	15% decimation	20% decimation	30% decimation
Target Threshold					
Vth1	--	0.38 (mV)	0.86 (mV)	1.52 (mV)	3.43 (mV)
Vth2	0.151 (V)	136.20	129.19	122.36	109.24
Vth3	--	166.05	173.96	182.04	198.74
Vth4	0.585	557.14	543.48	529.97	503.41
Vth5	--	613.23	627.61	642.14	671.66
Vth6	1.25	1,212.40	1,193.75	1,175.20	1,138.40
Vth7	--	1,287.98	1,307.10	1,326.32	1,365.02
Vth8	2.066	2.023 (V)	2.002 (V)	1,980.22	1,937.62
Vth9	--	2.109	2.130	2.152 (V)	2.195 (V)
Vth10	2.934	2.891	2.870	2.848	2.805
Vth11	--	2.977	2.998	3.018	3.062
Vth12	3.75	3.712	3.693	3.674	3.635
Vth13	--	3.786	3.806	3.825	3.862
Vth14	4.415	4.387	4.372	4.358	4.328
Vth15	--	4.443	4.457	4.470	4.497
Vth16	4.849	4.834	4.826	4.818	4.801
Vth17	--	4.864	4.871	4.878	4.891
Vth18	--	4.999	4.999	4.998	4.997
Reference Voltage	5.00(V)	5.00(V)	5.00(V)	5.00(V)	5.00(V)
Signal Input Range	0 to 5.0(V)	0 to 5.0(V)	0 to 5.0(V)	0 to 5.0(V)	0 to 5.0(V)

Table 10.(cont.) Testing Configulation of Parity Circuit (Threshold Voltage)

samples. The SNS ADC converts that signal to a thermometer code using the comparator ladder and changes the thermometer code to a binary code by means of a program logic array (PLA) such as an EPROM. The mod 9 comparator lines are also used to generate the parity. To process this information, a Hewlett Packard *model 1631A/D* logic analyzer is used. The model 1631A/D is a general purpose logic analyzer for performing state, timing, and analog waveform measurements. The output data and parity bit information are stored in the storage device (floppy disk) via the logic analyzer. The data for each decimation width is stored and compared. This information is also compared to the output of the MATLAB simulation program, sys_simu.m.

3. Testing Results

CRAIG7.VI creates a cosine square wave with a 5.0 V peak to peak voltage amplitude. Each target threshold voltage and resistance value is computed by MATLAB (Vth_calc.m, and idlprty.m). A combination of fixed resistors and potentiometers are used in the comparator resistance ladder to adjust the threshold level. The total resistance value of each ladder is 100 kΩ. The hardest part of building up the circuit is adjusting these threshold voltages. Adjusting one element in the ladder circuit to get a desired threshold voltage affects all other threshold values, so the remaining resistances need to be adjusted as well.

Adjusting threshold voltages for the parity circuit modulus were especially difficult since the circuit has pairs of resistance values with only a small difference. In addition, it has $2m_1$ potentiometers for the smallest channel of mod m_1. The logic analyzer is used to check all of the logic values from the SNS ADC. Figure 22 shows the comparator ladder output of mod 10 captured from the logic analyzer display. Clearly the output is symmetric. Labels B0 to B8 of the figure indicate each comparator output. B0 shows the bottom (or lowest) level comparator state, and B8 shows the top (or highest) level comparator state. After adjusting the resistance value to get the symmetrical output, the static analysis is performed. The output state, decimal number representation and parity bit information, are collected for each decimation width (10%, 15%, 20%, and 30%). Table 11 shows the parity circuit behavior of the 8-bit SNS ADC. MATLAB is used as a reference for the parity bit information. The parity "1" bit indicates a rejected sample proportional to the decimation width. In this test, 646 samples are used to express the 8-

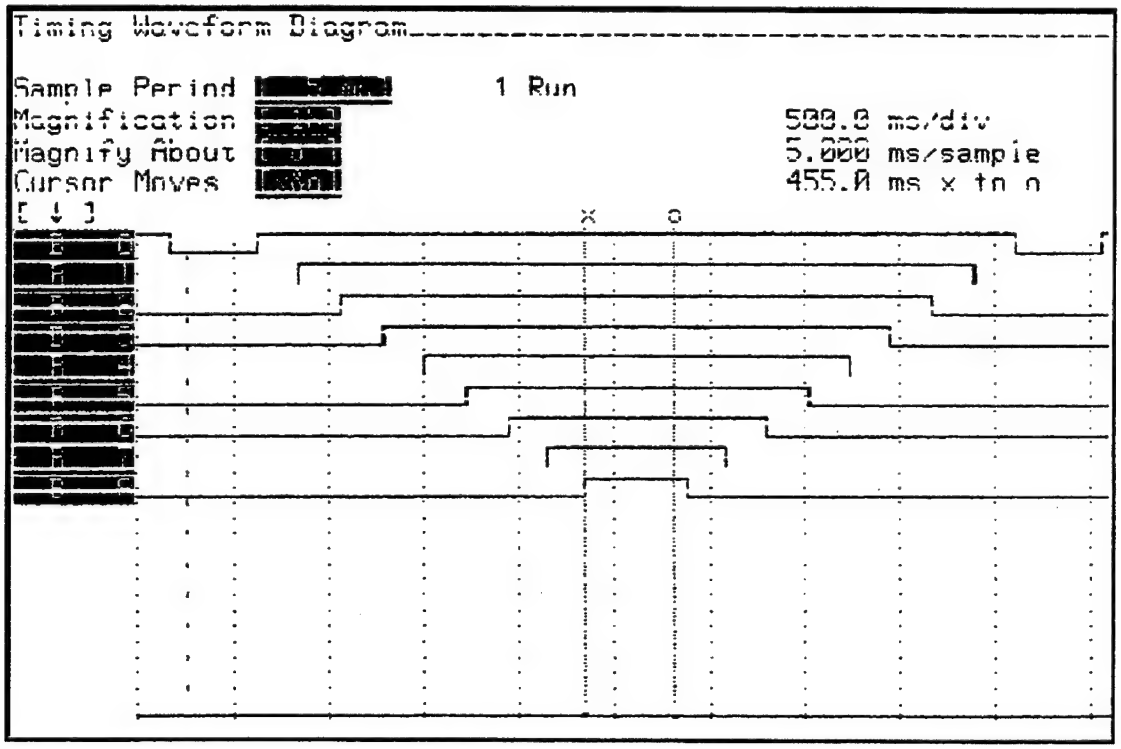


Figure 22. Timing Analysis of a Folding Circuit (mod 10)

	Total number of parity "1" from MATLAB simulation	Total number of parity "1" from the hardware	Number of matched parity"1" (MATLAB and Hardware)	Rate of the parity "1" samples relates to the number of the sample	Parity "1" recognition rates
10%deci-mation	65	130±20	50	20~23%	75~80%
15%deci-mation	96	150±10	70	22~25%	70~80%
20%deci-mation	129	180±10	120	25~30%	90~97%
30%deci-mation	195	245±10	190	35~40%	95%~

Table 11. Parity Circuit, Decimation and Hardware Performance Comparison

bits. As the decimation width increases, the total number of parity "1" bits increases, and the recognition rate, which shows how much the actual number of parity "1" bits agree with the MATLAB-predicted parity "1" bits, also increases. The larger the decimation width, the fewer number of output errors. However, the larger the decimation width, the higher the number of samples rejected. For example, in Table 11, 30% decimation gives a recognition rate in excess of 95%, but over one third of the total samples are rejected. From Table 11, it can be said the rate of rejected samples versus the total number of samples remains almost constant for decimation widths less than 20%, and recognition rates remain constant till the 15% decimation width.

The MATLAB file errormod.m is used as a comparison tool for "before" and "after" corrections. The logic analyzer model 1631A/D stores the sample index, output state (decimal), and the parity bit data. The file errormod.m checks these three sets of information, and it regards the data as corrected when the output state error and parity bit "1" occur at the same time. If the output error occurred, and the parity bit was "0", then MATLAB regards the error as uncorrected. This means the system put out erroneous information. Obviously a higher error correction rate is better. Table 12 shows the experimental results of error correction rates. Each decimation width case is run three times to collect data. The larger decimation width correlates with a better error correction rate.

Figures 23 through 26 show the results of error correction for each decimation width. For each case, the errors present decrease significantly, and a 20% decimation width is enough to get a 100% error correction. In these figures, the x-axis shows the sample index which relates to the input voltage of each MZI, and the y-axis shows the decimal output state. Some of the errors are within the 8-bit state (less than 256). Others are out of the 8-bit state (greater than or equal to 256). Figure 27 shows the relationship of error correction and decimation width graphically. The results are summarized as follows:

- The larger the decimation width the higher the recognition rate of the expected parity bit.
- Larger decimation widths yield higher error correction rates.
- Larger decimation widths may cause excessive problems such as high rates of rejecting sample output.

20% decimation width is recommended as the best choice for the parity circuit configuration.

%Decimation	Total number of error	Total number of corrected error	Error correction rates (%)	Average error correction rates
10	18	16	88.8	84.1
10	21	17	80.1	
10	24	20	83.3	
15	18	18	100	89.8
15	24	23	95.8	
15	18	16	88.8	
20	28	18	100	100
20	37	18	100	
20	30	22	100	
30	23	23	100	100
30	19	19	100	
30	23	23	100	

Table 12. Error Correction Data of 8-bit SNS ADC

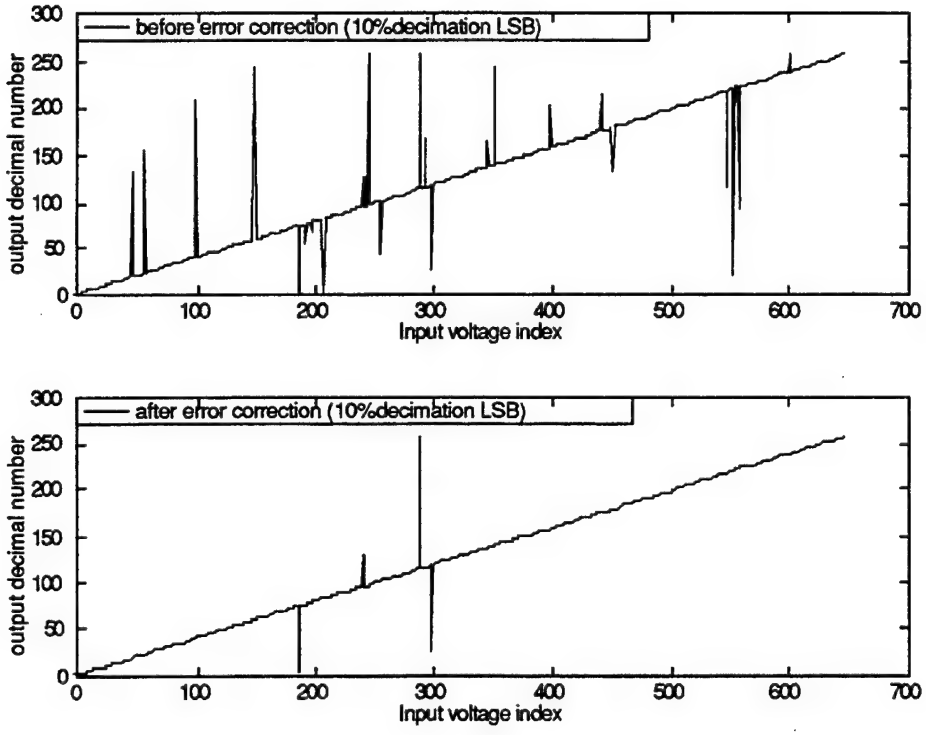


Figure 23. Characteristic Analysis of an 8-bit SNS ADC (10%Decimation)

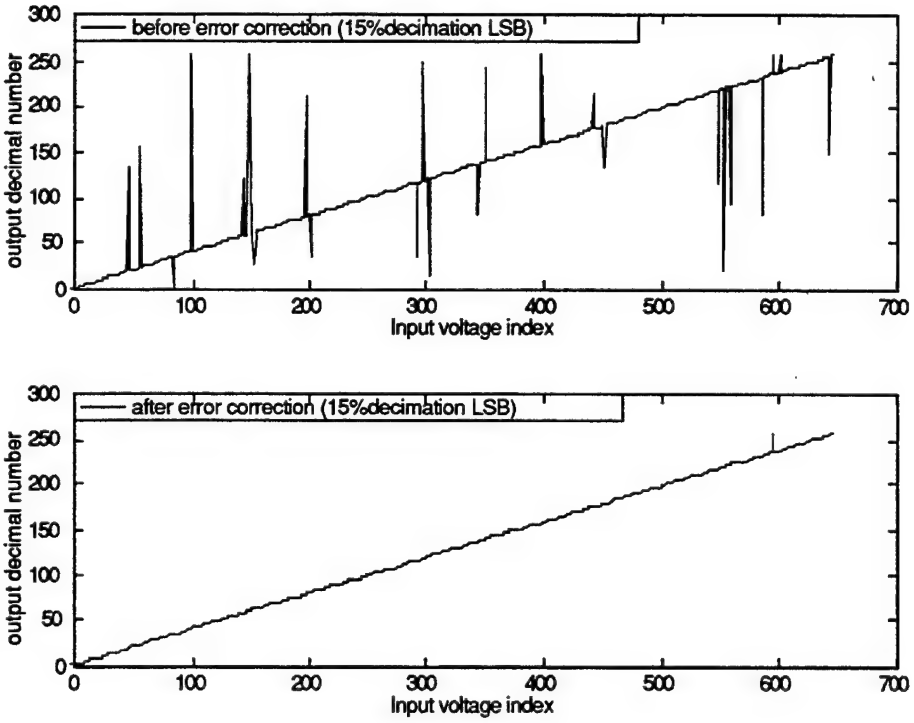


Figure 24. Characteristic Analysis of an 8-bit SNS ADC (15%Decimation)

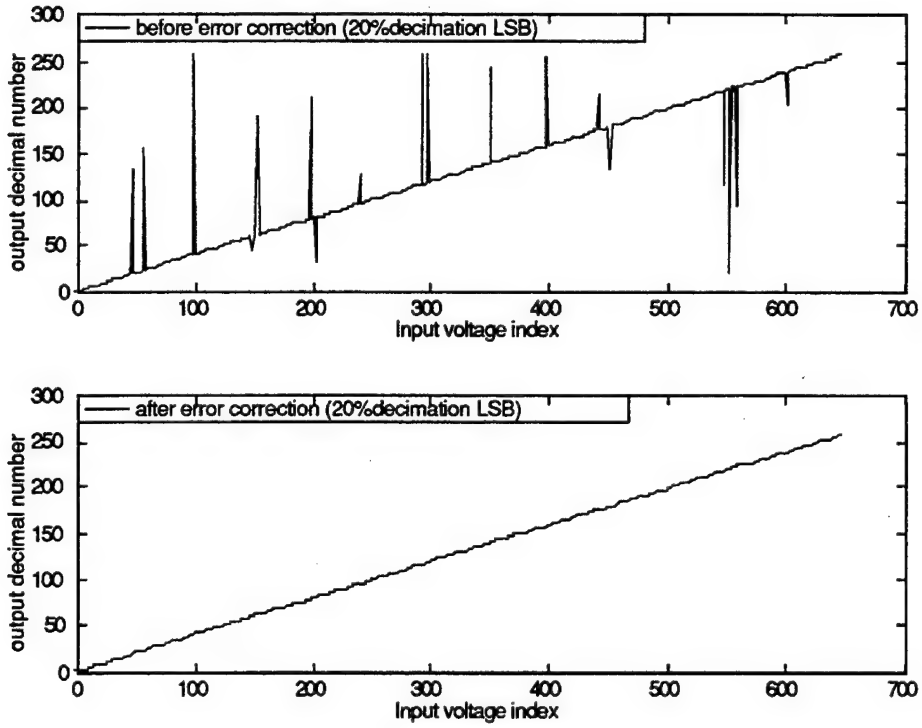


Figure 25. Characteristic Analysis of an 8-bit SNS ADC (20%Decimation)

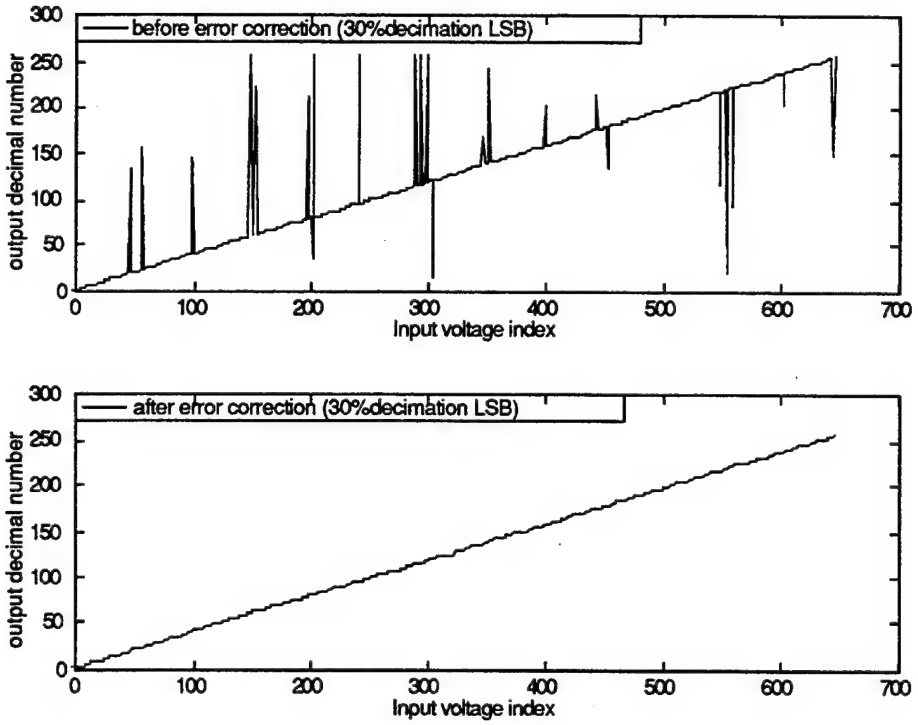


Figure 26. Characteristic Analysis of an 8-bit SNS ADC (30%Decimation)

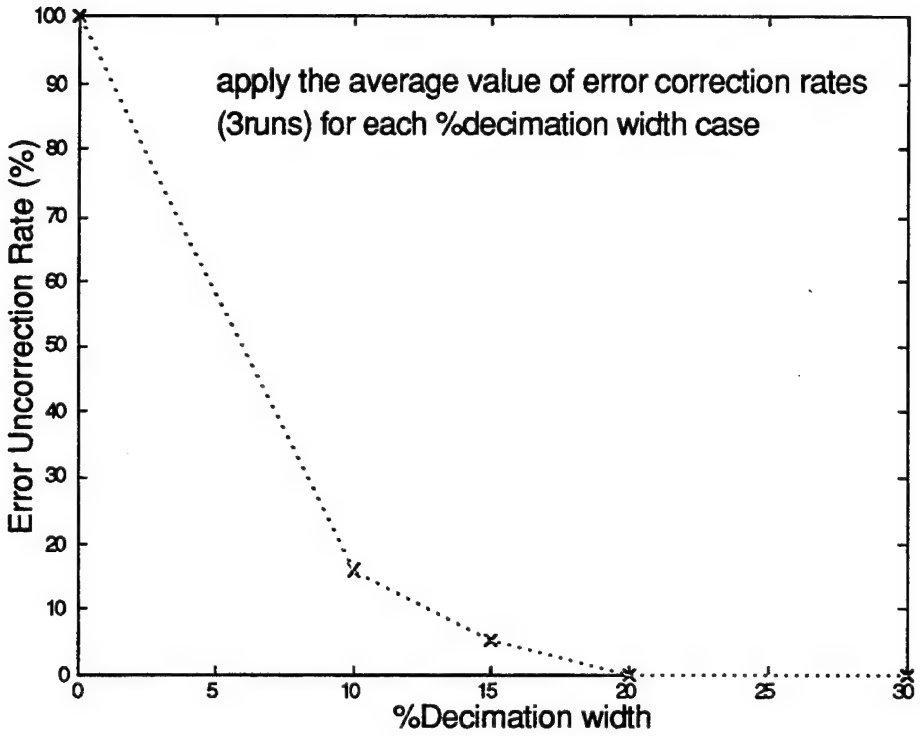


Figure 27. Characteristic Analysis of an 8-bit SNS ADC

B. TESTING AND EVALUATING THE TIMING PROCESSOR

1. Signal Performance

The timing processor is built as discussed in Chapter III. A picture of the timing circuit is shown in Fig. 28. The circuit is made by using a wire wrapping board. Supply voltages (+12 V, ±5 V) and ground for the circuit are applied. The input signal and the output of each stage can be detected from the BNC terminal in this circuit. Two DIP switches are shown on the circuit. One is for simulating the EPROM output signal, and the other is used as an address line for the programmable delay units, PDU-16F. In order to cool down the chip, the regulator LM7805CT has a heat sink. Capacitors are added for each chip to terminate the high frequency effects. A Digitizing Oscilloscope TDS420 is used to capture the waveform of the timing processor. The TDS420 can take snapshots of the four individual waveforms. Each terminal can choose a 50 Ω or 1 MΩ terminal. The TDS 420 stores the image in the personal computer via a GPIB bus. Also the TDS420 can handle a 100 mega sample per second (MSPS) maximum digitizing rate, and a 150 MHz maximum analog bandwidth.[Ref. 22] Figure 29 shows the output waveform of the photo detector, model 1811. The laser transmitter model 400 applies a 20 nsec pulse with a pulse repetition interval (PRI) of 200 nsec in this figure (10% duty cycle). The figure shows that the detected pulse width is almost the same as the pulse width of the laser transmitter, which means the propagation delay is minimal. The figure also shows that the highest value of the photo detector is 146.0 mV, and the average noise layer value is around 15 mV. This input signal, however, changes its waveform when the photo detector output connects to the input terminal of the timing processor. Figure 30 shows the signal waveform at the input terminal of the timing processor without applying power. Instead of the nice pulse shape of Fig. 29, there are some higher order frequency effects in the waveform. This waveform also changes when supply voltage is applied. Figure 31 shows the signal waveform at the input terminal of the timing processor after supplying the power. The waveform in Fig. 31 clearly shows multiple peaks. The maximum measured voltage of the main pulse is 360 mV, but some of the other pulses have voltage spikes in excess of 200 mV. This input signal passes through the AD9698 comparator. However, it is hard to set up the threshold voltage of the comparator to get the main pulse only, since the main

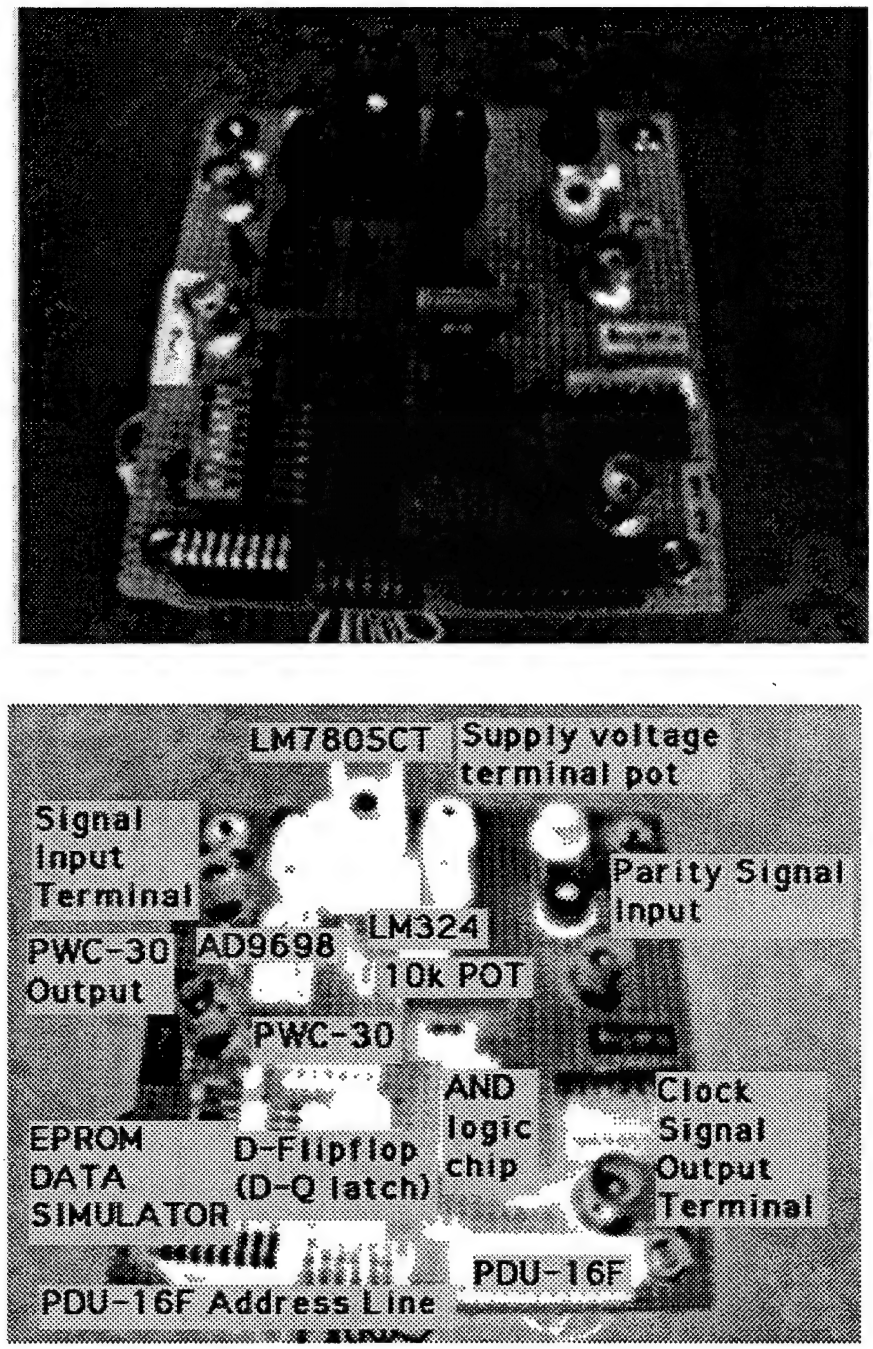


Figure 28. Picture of the Timing Processor

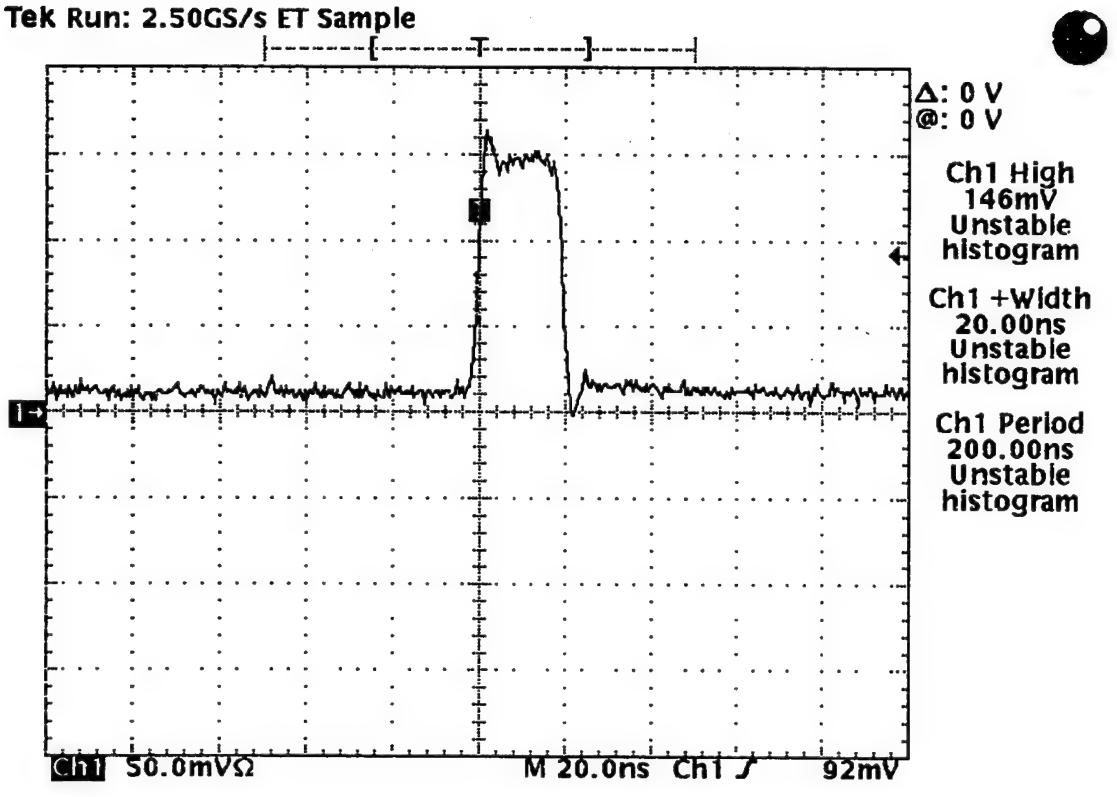


Figure 29. Output Waveform of the Photo Receiver model 1811(New Focus, Inc.)

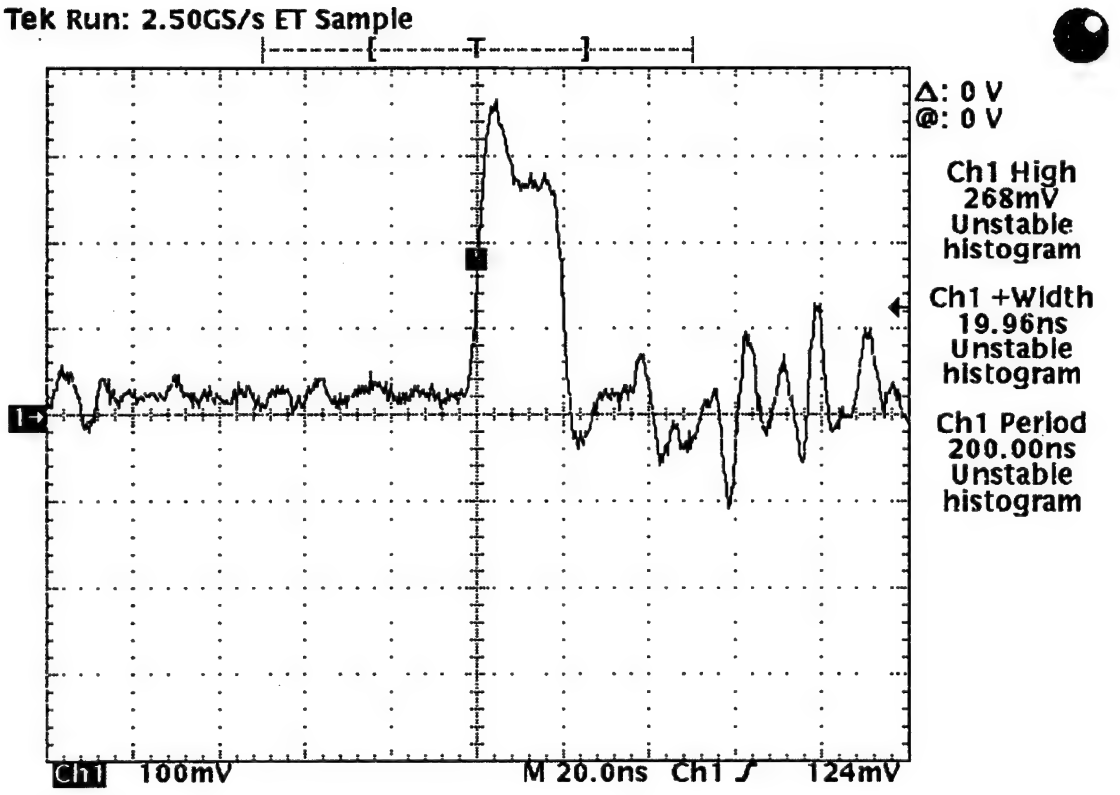


Figure 30. Input Signal Waveform of the Timing Processor Without Supply Voltage

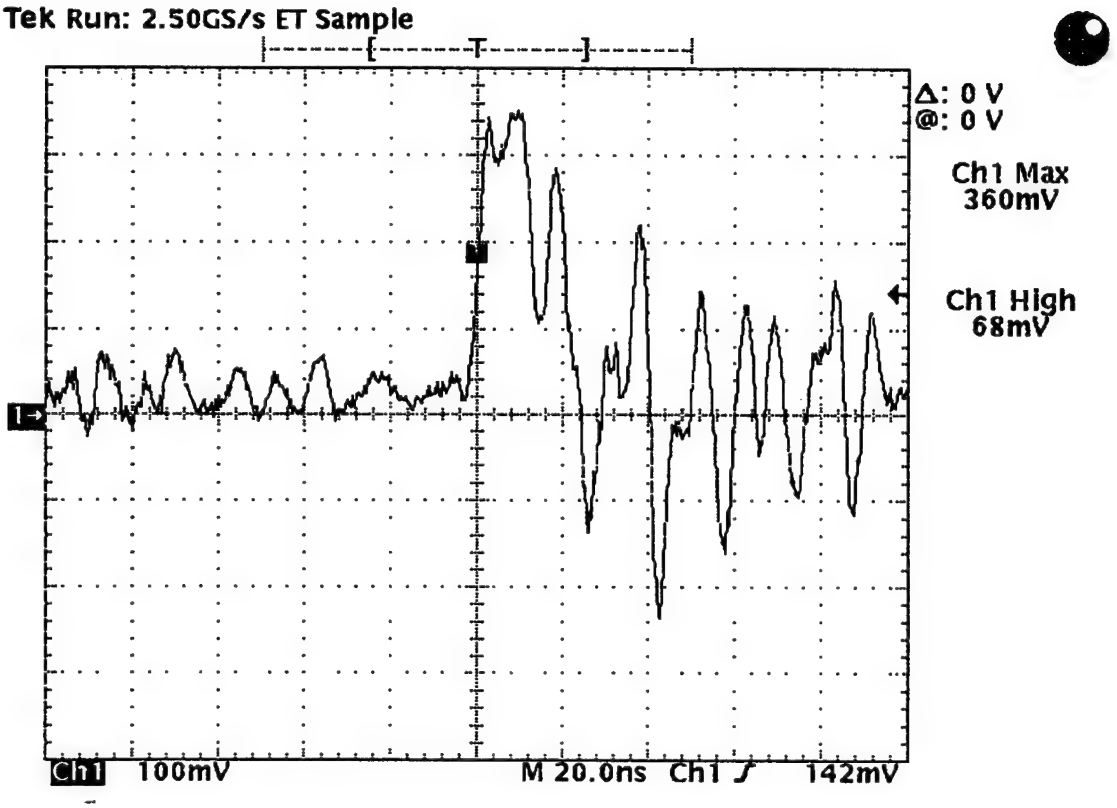


Figure 31. Input Signal Waveform of the Timing Processor With Supply Voltage

pulse has a dip whose value is close to the peak value of the second pulse. This causes multiple triggering of the comparator, and consequently a plural TTL level output from the comparator. The output waveform of the comparator AD9698 is shown in Fig. 32. The AD9698 generates three sharp pulses because of the multiple triggers as shown in Fig. 31. These pulses have a TTL level output and also are very sharp. Each pulse has a 5 to 6 nsec pulse width, so this usually causes a misfire of the clock and a big timing problem. This effect, however, is diminished by making use of the pulse width controller PWC-30. As noted in Chapter III, the PWC-30 has a very effective feature in that it activates using a rising edge trigger, and no other pulse can retrigger the device during the operation. Figure 33 shows the characteristics of the PWC-30 in operation. This figure explains how the PWC-30 works for various inputs. The signal output of the PWC-30 is independent of the pulse width of the input signal and each signal output has a fixed pulse width τ. The waveform difference of the PWC-30 input / output is shown in Fig. 34. The upper signal is the input to the PWC-30 and this waveform is almost the same as the one in Fig. 32. The lower waveform is the PWC output and this pulse has a 50 nsec pulse width. By measurement, the propagation delay of the PWC-30 is around 7 nsec which is within the specifications. The output signal shows two dips in the pulse,but these local minima are both above the "ON" level, and therefore do not affect other parts of the timing processor. The 50 nsec pulse signal generated by the PWC-30 and the parity circuit output both go through the AND logic (74S08). Finally, the clock pulse of the D-Q latch is obtained. The timing processor and folding circuit could not be integrated at this time, so a simulated parity signal was used in testing instead of the actual parity signal. In the following section, the simulated parity signal is generated by WAVETEK model 145 or by toggling the output of the PWC-30. WAVETEK model 145 can generate a TTL level pulse signal at frequencies up to 20 MHz.

2. Timing Analysis of the Processor

The next step in the evaluation of the timing processor is to investigate the timing relationship for each stage of the processor, and to confirm the operation of the timing processor. Generally, each IC chip has a certain propagation delay time during the signal I/O. Even the fastest devices experience these delays. In the timing processor, several chips are used, and it is important to know the total system time delay. If the time delay is longer than the PRI, then the sampling is not working correctly. The system does not

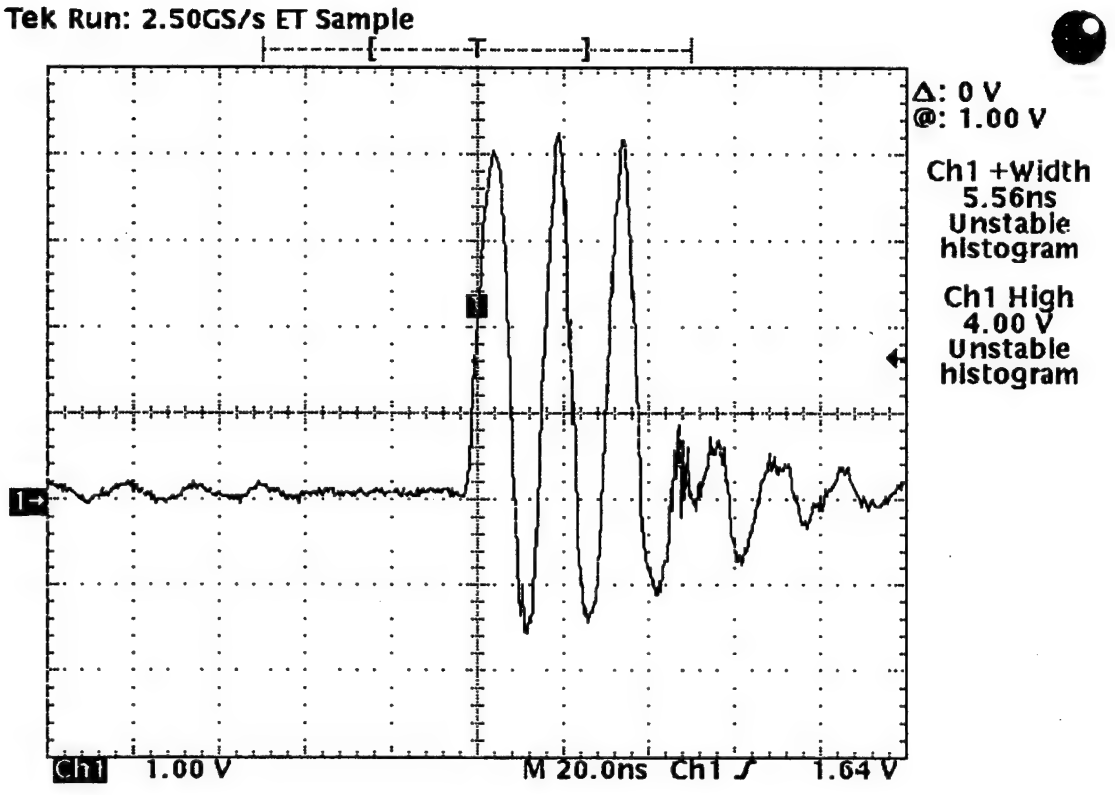


Figure 32. Output Signal Waveform of Comparator AD9698 in Timing Processor

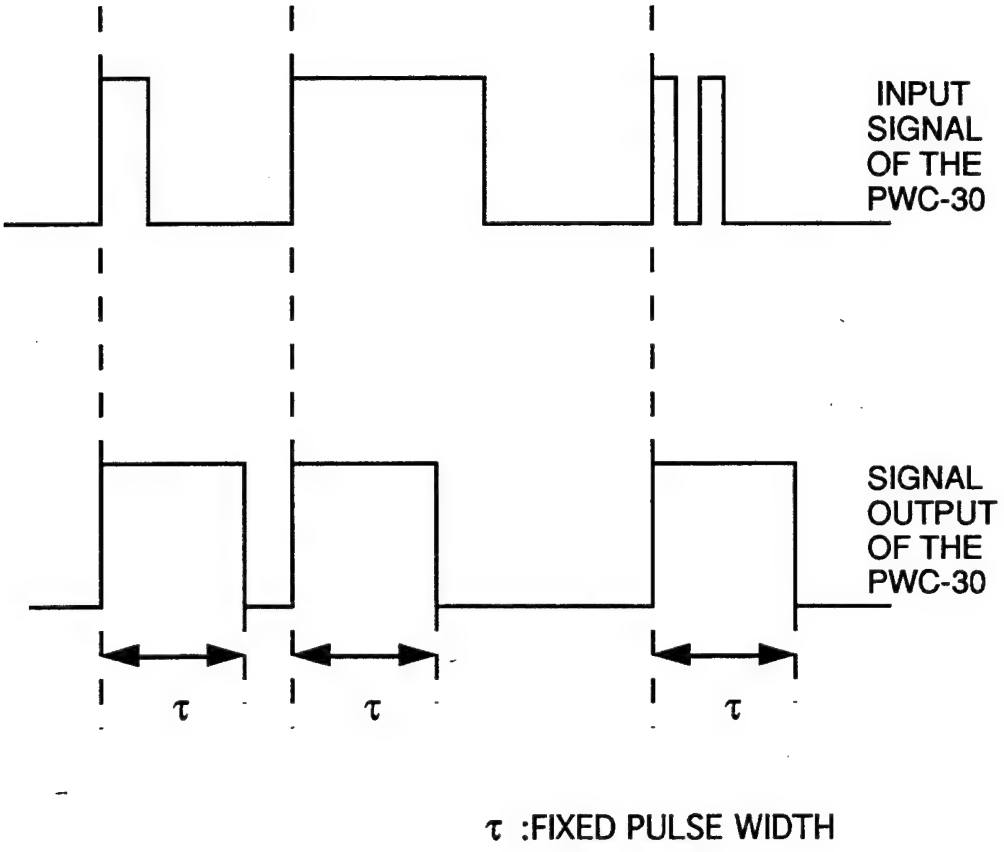


Figure 33. Operation Characteristics of the PWC-30

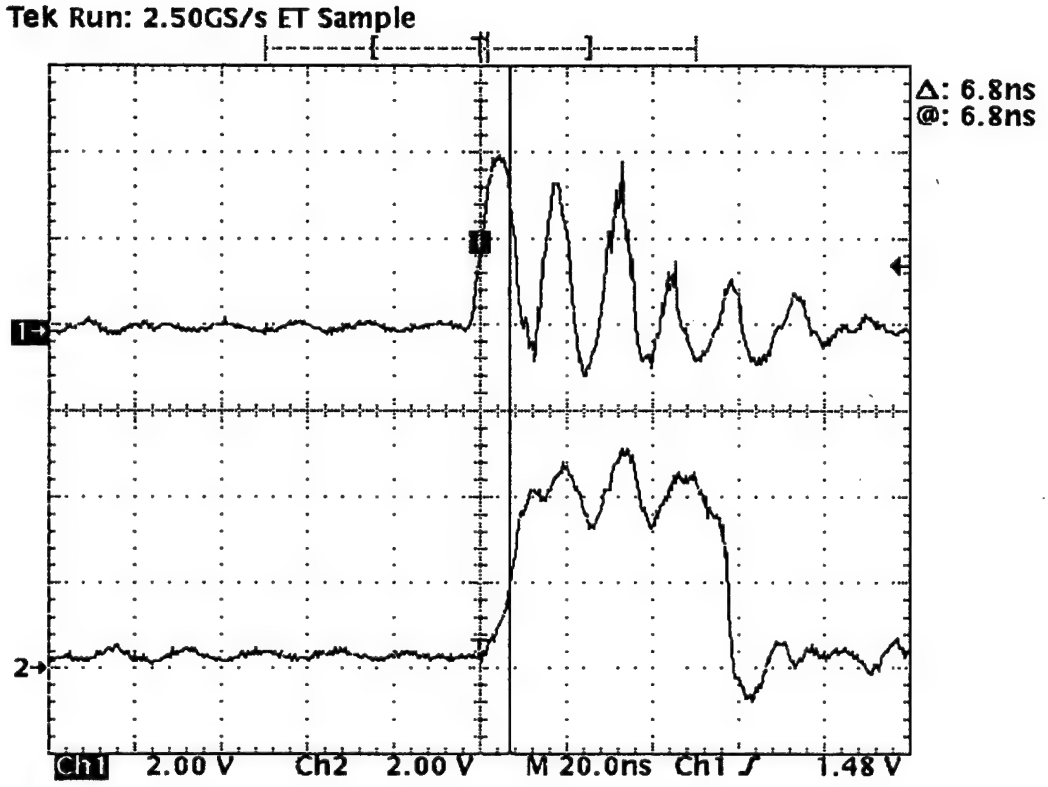
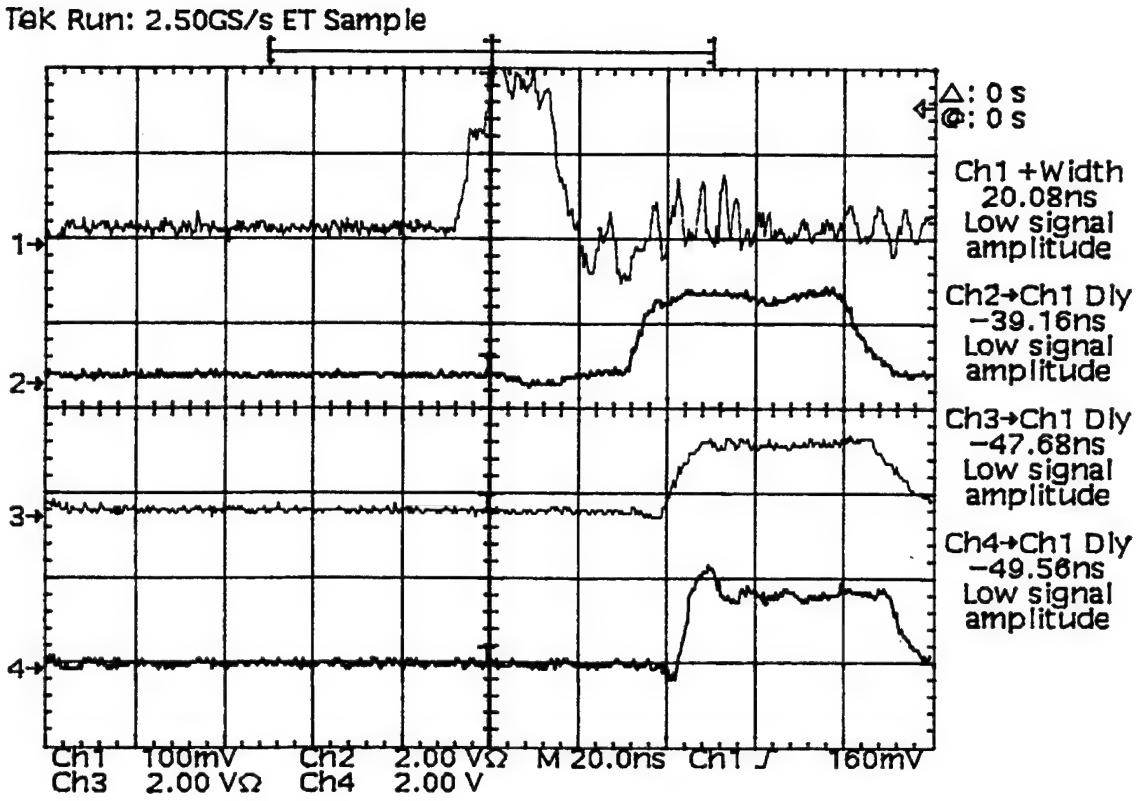


Figure 34. Waveform Difference of the PWC-30 Input / Output

work because the timing clock does not match the sample. A similar problem occurs when the time delay is too short compared to the data. However, we want to avoid the former case rather than the latter case because the data bus has many lines. Correcting for the time delay in the latter case would require only one delay line for the time delay. Figure 35 shows the clock pulse waveform of each level. Channel 1 shows the input signal to the system. The rising edge of this signal is used as the reference point of the propagation time delay analysis. Channel 2 shows the output of the PWC-30. Channel 3 has the same signal path, but it is measured at the AND logic input. The last channel shows the output of the AND logic which is the actual clock signal of the D-Q latches. In this case, the output signal of the PWC-30 is applied to both 74S08 input pins as a timing clock pulse and a simulated parity check signal. The right side of the figure shows the delay time of each channel compared to channel 1. According to the figure, the total delay time from photo detector to the final clock output is 50 nsec. If we obtain the propagation time delay of each component from the specification data, the total delay time should be around 20 nsec. The timing delay, however, can be affected by the wiring. The biggest cause of the delay is the signal line from the photo detector to the timing processor since a very long coaxial cable is used in this experiment. Even the short length of wire used in the wire wrapping can contribute to the total delay. For example, in an ideal circuit, the pulses in channels 2 and 3 should be coincident and have the same timing delay compared to the photo detector output. However, Fig. 35 shows an 8.5 nsec pulse delay between the two devices in the actual circuit. The final clock pulse has a TTL level output, and its pulse width is 50 nsec. This signal arrives at the clock input pin of D-flip flop 74ALS374, and proceeds to update the data output. In this configuration, the PRI of the system is set at 200 nsec, so the timing process will eliminate the mismatching of sampling data and timing signal at the D-flip flop. This timing process is also analyzed using the simulated parity bit signal, and simulated data bus information. The Hewlett Packard logic analysis system *model 16500B* is used for timing analysis. The model 16500B has much more capability than the model 1631A/D logic analyzer used in the decimation error testing. Model 16500B has some features not included in the model 1631A/D such as:

- 100 MHz state and 500 MHz timing acquisition speed
- 96 data channels / 6 clocks
- GPIB capability (programming, data stored)



The waveform for each channel indicates the following signal

Channel 1 : the input signal to the system (from the photo detector)
Channel 2 : the output waveform of the PWC-30
Channel 3 : the input waveform of the 74S08 (same path of the Ch. 2)
Channel 4 : the output wave form of the 74S08

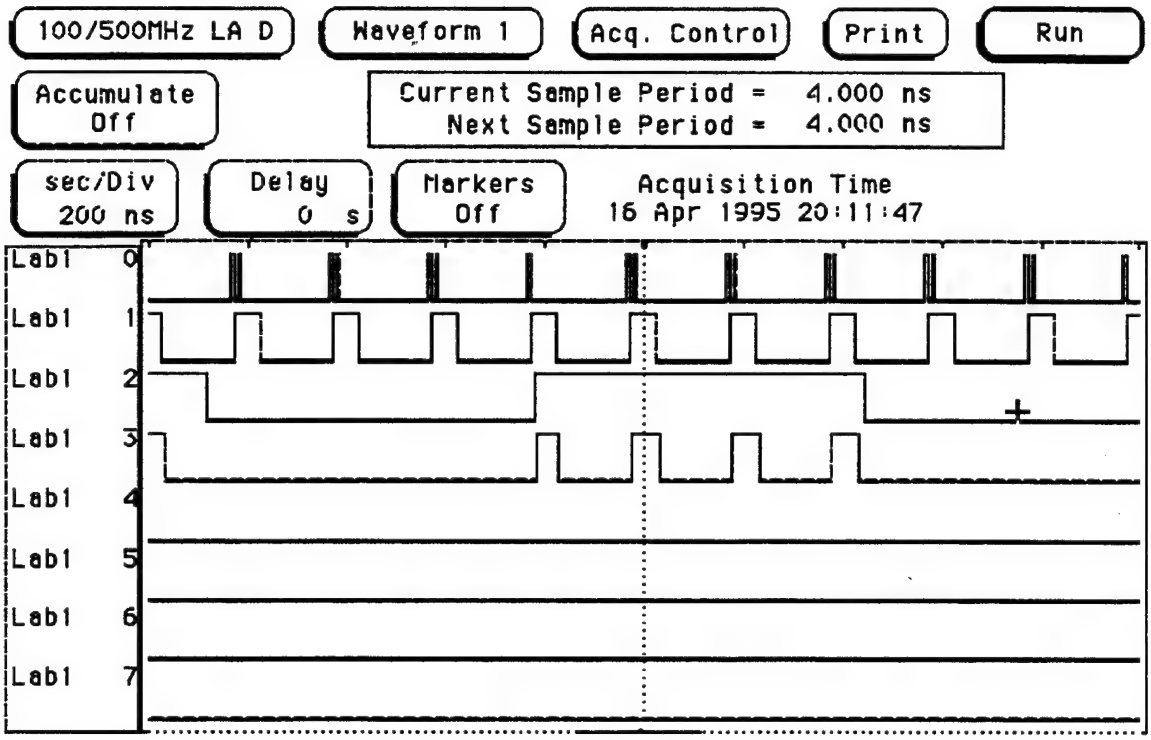
Figure 35. Waveform of Each Stage of the Timing Clock Signal

- 3.5 nsec window

This equipment is mainly used for SNS ADC folding circuits analysis.[Ref. 23] The WAVETEK model 145 is used as a simulated parity bit signal. Figure 36 shows the display of the model 16500B. The index numbers 0 to 3 are used. Each signal is a TTL level output, and each index shows the output of the comparator AD9698, the output of the pulse width controller PWC-30, the simulated parity bit signal from WAVETEK model 145, and the final clock pulse to the D-flip flop. The laser pulse configuration is the same as the previous experiment with a 20 nsec pulse width and a PRI is equal to 200 nsec. The model 145 generates a 700 kHz square wave with a 50% duty cycle to confirm the operation of the AND logic. The multiple triggering of the comparator is seen on the index 0 as mentioned in the previous section. The pulse width controller, however, eliminates this problem, and generates a periodic pulse train with a 50 nsec pulse width. The 74S08 AND logic compares the clock input from the PWC-30 with parity bit information, and the final output is the exact pulse waveform that we predict. Another example of the timing analysis is shown in Fig. 37. The model 145 generates a 5 MHz pulse with a 50% duty cycle as a parity bit signal. Index 5 shows the output waveform of the programmable delay unit, PDU-16F. The pulse delay unit maintains a certain time delay and pulse width from the 74S08 output. For the programmable delay unit, a DIP switch is used to enable each address line to change the delay time. The fixed delay time is explicit in the figure. In this example, address line number 5 is in the ON state, and the OFF state signal is applied to other address lines. According to the specifications for the programmable delay unit in Table 8, the inherent delay on pin 1 is 8 nsec, and the incremental delay ΔT_d is 2 ± 0.5 nsec per step. Table 13 shows the truth table of the PDU-16F. According to Table 13, the configuration of this example produces a delay output T_{16}. The relationship between the step number n_d and the time delay T_d is

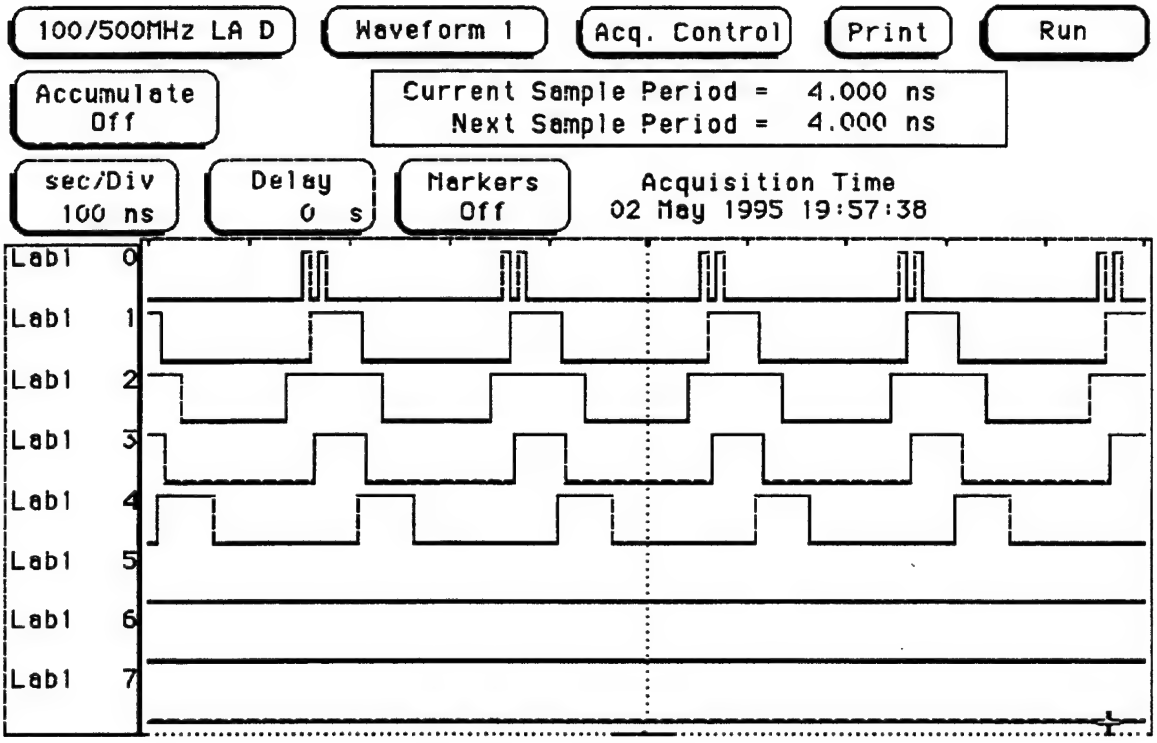
$$T_d = T_{DO} + \Delta T_d \cdot n_d \tag{26}$$

Therefore the T_d of a delay output T_{16} is 40 ± 8 nsec. By observation of Fig. 37 using the timing cursor, the time difference between two signals (input / output of PDU-16F) is 44.0 nsec. This satisfies the specifications, and confirms that the PDU-16F is operating



Each label shows the following TTL level output :
Label 0 : the output signal of the AD9698
Label 1 : the output signal of the PWC-30
Label 2 : the simulated parity bit signal (WAVETEK 145)
Label 3 : the output signal of the 74S08
(Label 4 to 7 : unconnected)

Figure 36. Display of the Logic Analyzer Model 16500B



Each label shows the following TTL level output :
Label 0 : the output signal of the AD9698
Label 1 : the output signal of the PWC-30
Label 2 : the simulated parity bit signal (WAVETEK 145)
Label 3 : the output signal of the 74S08
Label 4 : the output signal of PDU-16F
(Label 5 to 7 : unconnected)

Figure 37. Another Example of the Timing Signal Analysis

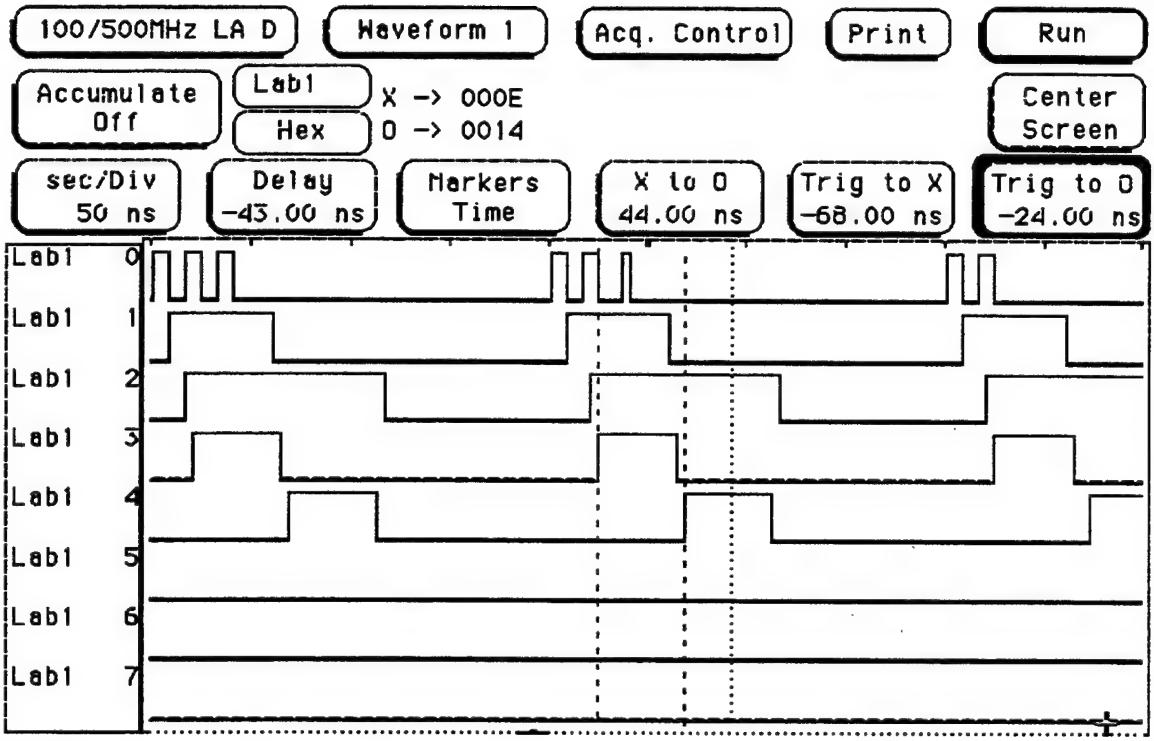
Delay Out	Address#6	Address#5	Address#4	Address#3	Address#2	Address#1
T_0	0	0	0	0	0	0
T_1	0	0	0	0	0	1
T_2	0	0	0	0	1	0
T_3	0	0	0	0	1	1
T_4	0	0	0	1	0	0
T_5	0	0	0	1	0	1
T_6	0	0	0	1	1	0
T_7	0	0	0	1	1	1
T_8	0	0	1	0	0	0
T_9	0	0	1	0	0	1
T_{10}	0	0	1	0	1	0
T_{11}	0	0	1	0	1	1
T_{12}	0	0	1	1	0	0
T_{13}	0	0	1	1	0	1
T_{14}	0	0	1	1	1	0
T_{15}	0	0	1	1	1	1
T_{16}	0	1	0	0	0	0
T_{17}	0	1	0	0	0	1
•	•	•	•	•	•	•
•	•	•	•	•	•	•
•	•	•	•	•	•	•
T_{30}	0	1	1	1	1	0
T_{31}	0	1	1	1	1	1
T_{32}	1	0	0	0	0	0
T_{33}	1	0	0	0	0	1
•	•	•	•	•	•	•
•	•	•	•	•	•	•
•	•	•	•	•	•	•
T_{62}	1	1	1	1	1	0
T_{63}	1	1	1	1	1	1

- 0 : OFF state 1 : ON state
- Don't care condition if enable (E_0) is ON

Table 13. Truth Table of the PDU-16F Address Line

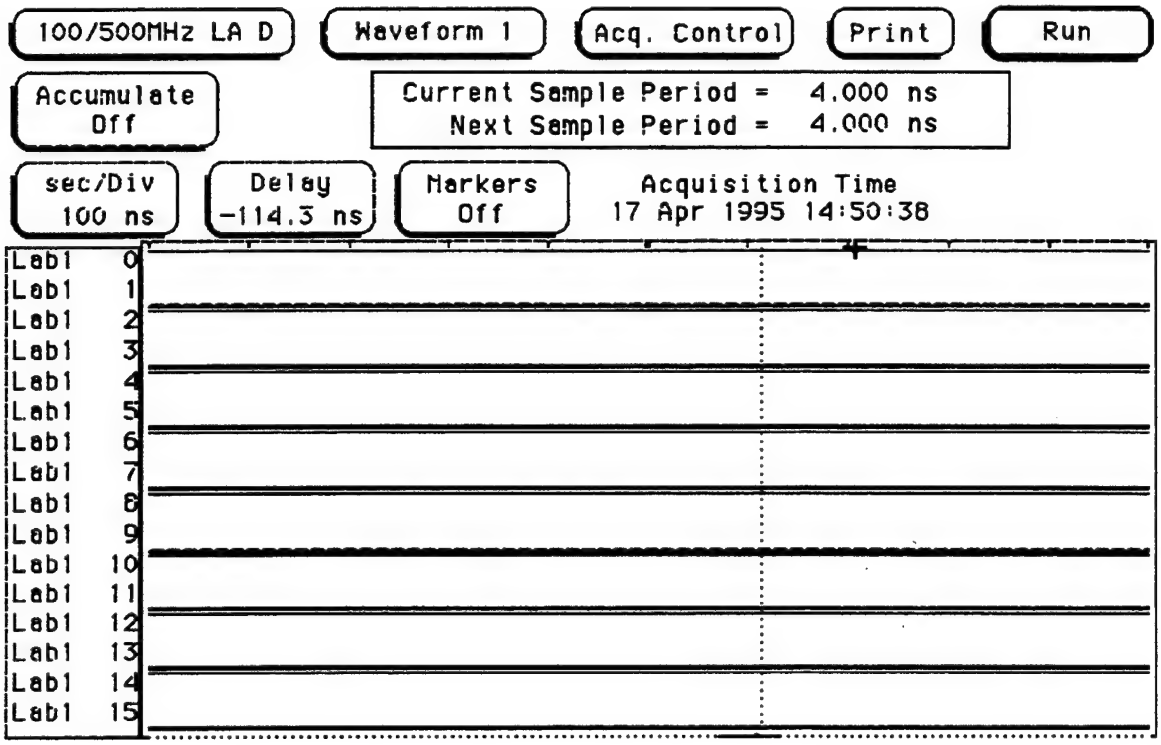
correctly. Figure 38 shows the timing signal behavior. The propagation delay of each stage in TTL level logic is measured using the same manner as in the previous example. Figure 37 differs from Fig. 36 in that the rising edge of the parity bit information and the PWC-30 output are not totally overlapped. Therefore, the 74S08 output has a shorter pulse width. This is, however, closer to the actual signal behavior because the actual parity bit signal has a narrower pulse width than the simulated parity bit signal. Using the two cursors shown in the figure, we measured the propagation delay with the results listed in Table 14. The first row in this table shows the propagation delay of the PWC-30. The second row indicates the propagation delay of 74S08.

Finally the simulated EPROM output data is applied to the 74ALS374 octal D-flip flop and the input / output of 74ALS374 is observed. The data output states are controlled by a dip switch, and this data signal is static. Each state signal is a DC output and not a high frequency signal output. Figure 38 shows one set of results. Index labels 0 to 7 show the data input of the 74ALS374, and index labels 8 to 15 show the data output of the 74ALS374. The DIP switch is set to generate the binary word "10101010". The word is applied to the input of the 74ALS374. The output should be the same binary word "10101010", and the same information does in fact come out for each corresponding pin. Because of the static input, the data for transient I/O behavior in the circuit cannot be recorded, and characteristic analysis using optical pulse signal is needed for further research.



Each label shows the following TTL level output :
Label 0 : the output signal of the AD9698
Label 1 : the output signal of the PWC-30
Label 2 : the simulated parity bit signal (WAVETEK 145)
Label 3 : the output signal of the 74S08
Label 4 : the output signal of PDU-16F
(Label 5 to 7 : unconnected)

Figure 38. Details of the Timing Processor Signal Behavior



Each label shows the following TTL level output :
Label 0 to 7 : the simulated EPROM data input
(the data "10101010" is received)
Label 8 to 15 : the data output from 74ALS374
(the data "01010101" is sent)

Figure 39. Data In/Out Analysis of the Timing Processor

Rise Edge Reference A	Rise Edge Reference B	Propagation Delay
AD9698 output	PWC-30 output	8.0 (nsec)
PWC-30 output	74S08 output	4.0 (nsec)
74S08 output	PDU-16F output	44.0 (nsec)*

* Choose Delay Out as T_{16} and this satisfies the specification

Table 14. Observation Results of Propagation Delay

V. SUMMARY AND FURTHER RESEARCH

This thesis provides details on the system simulation of an 8-bit SNS ADC system. The simulation results are used to compare the performance of the hardware circuits with the ideal performance. Also, the design and analysis of a parity timing processor is given. The timing processor is built, and the clock signal of the processor using the detected optical pulse signal input is analyzed. The MATLAB program developed in this thesis can easily be applied to other SNS ADC configurations using different moduli.

One of the goals initially envisioned for this thesis was to characterize the whole SNS ADC component. This, however, had to be changed because of the need to integrate the folding circuit and timing processor. Therefore, this thesis discusses only the static data for the timing processor characteristics. For future research, signal analysis of the data bus and timing processor are needed to observe the time difference between both signal paths. Also, timing analysis of the data I/O at the latching states using the optical pulses is needed.

APPENDIX MATLAB™ SOURCE CODE

A. Vth_calc.m

```
% Thesis program No.1 (Vth_calc.m)
% This m-file generates normalized threshold intensity / voltage
% and its difference between the neighbors. In this program a
% number of moduli X is needed as an input.
%                                          Written by Hiromichi Yamakoshi

clc,clg                                    % input the number of modulus, X
X = input('What number of modulus do you want? Please input number : ');
Drange =X*(0:1/X:1);                       % creates input voltage index
                                           % from 0 to X
output = (sin((pi*Drange)/(2*X))).^2;      % apply Eq. 3
n = 1:length(output)-1;
diffD = diff(output);                      % calculate the difference between
                                           % each two normalized threshold
plot(Drange,output,Drange,output,'o'),grid % start plotting process
title(['Calculation of Normalized Threshold in modulus ',num2str(X)])
xlabel('Input Voltage Index')
ylabel('Normalized Detector Output')
diffD =[0 diffD];                          % display normalized threshold
disp(['Vth of mod',num2str(X),' dVth of mod',num2str(X)])
disp([output' diffD'])
```

B. thmtobin.m

```
% Thesis program No.2 (thmtobin.m))
% This m-file generates thermometer code and converted binary code with input of
% desired number of modulus X.
%                                          Hiromichi Yamakoshi
```

```
clc
X = input('What number of modulus do you want? Please input number : ');
x = X;                                  % First, using X, create bit information
n = 1;
while  x > 2
       x = 0.5 * x;
       n = n+1;                         % n : bit information
end
disp(['We need  ',int2str(n),'  bits.']);
disp('and')
disp(['possible comparator information of moduli  ',int2str(X),'  is'])
comparator = zeros(X);                  % Next, create the thermometer code
for i= 1:X                              % comparator : thermometer code info.
       comparator(i,:) =[ones(1,i-1), zeros(1,X-i),i-1];
end
compout = comparator(:,1:X-1);          % reshape the thermometer code output
state = comparator(:,X);
disp(rot90(comparator))
disp('  cf.   the first row number shows comparator states.')
bin = zeros(X,n);                       %finally get the binary code
 for i = 1:X;                           % to create binary code, we uses the index
       m = 1;                           % information X
       p = i-1;                         % p creates index starting 0 to X-1
       if p == 0                        % create bin. information "0"
              bin(i,:)=bin(i,:);
       else                             % create bit information "1"
              while p >=2
                     q = rem(p,2);
                     bin(i,n+1-m) = q;
                     p = fix(p/2);
                     m = m+1;
              end
```

```
        bin(i,n+1-m) = 1;
        end
end
bin0 = fliplr(bin);
binout = [rot90(state);rot90(bin0)];    % reshape the binary code output
disp('also')
disp('binary information of each state is')
disp(binout)
disp('  cf.   the first row number shows comparator states.')
```

C. Lab_simu.m

```
% Thesis program No.3(Lab_simu.m)
% This m-file generates the number of folds and thermometer code of each sample
% point of each modulus, the number of sample point between two thresholds of
% each comparator, and predicted normalized LabVIEW output of each sample
% point.
% This program is designed for 8-bit SNS ADC LabVIEW circuit
%
%                                                    Hiromichi Yamakoshi

% First, we input LAB VIEW setting information.

total_index = 5000;          % 5000 samples are used as the total index.
pt_cycle9  = total_index/110; % 110, 99, and 90 are used as number of cycle of
pt_cycle10 = total_index/99;  % each modulus, and generates the number of
pt_cycle11 = total_index/90;  % sample point per cycle of each modulus.

                             % using the info above, generates the number of
pt_state =  pt_cycle9/(2*9);  % sample point between two thresholds of each
                             % comparator
```

```
% now input the number of sample points of observation. Now we choose modulus
% 9,10 and 11 for 8 bit SNS ADC. We use 256 dynamic range states out of 990.
% So we actually need the index of sample number, 0 to 646

index=0:646;
mod9folds  = fix(index/pt_cycle9);     % generates number of folds for each
mod10folds = fix(index/pt_cycle10);  % modulus relates to the sample number of
mod11folds = fix(index/pt_cycle11);  % the index which makes 8 bit resolution

% next we number the ststes of each modulus. Each modulus has the
%number which index from 0 to 2*(number of modulus) -1 in one cycle
% (Ex.) they have 18 states ( 0 to 17) in mod 9 (mod 9 has 9 states)

mod9states  = fix((index-(mod9folds*pt_cycle9))/pt_state);
mod10states  = fix((index-(mod10folds*pt_cycle10))/pt_state);
mod11states  = fix((index-(mod11folds*pt_cycle11))/pt_state);

Drange = mod9folds*(2*9) + mod9states;     % generates dynamic range

% Next, we create the thermometer code-expression of each modulus (i,e,
% 001111111 for mod 9, and so on)
M1=9;, M2=10;, M3=11;
code1=zeros(length(mod9states),M1-1);
code2=zeros(length(mod10states),M2-1);
code3=zeros(length(mod11states),M3-1);
% for mod 9
for i=1:length(mod9states);
        if mod9states(i) < M1
          code1(i,:)=[zeros(1,M1-mod9states(i)-1) ones(1,mod9states(i))];
        else
          code1(i,:)=[zeros(1,mod9states(i)-M1) ones(1,2*M1-mod9states(i)-1)];
        end
```

```
end
% for mod 10
for i=1:length(mod10states);
        if mod10states(i) < M2
          code2(i,:)=[zeros(1,M2-mod10states(i)-1) ones(1,mod10states(i))];
        else
          code2(i,:)=[zeros(1,mod10states(i)-M2) ones(1,2*M2-mod10states(i)-1)];
        end
end
% for mod 11
for i=1:length(mod11states);
        if mod11states(i) < M3
          code3(i,:)=[zeros(1,M3-mod11states(i)-1) ones(1,mod11states(i))];
        else
          code3(i,:)=[zeros(1,mod11states(i)-M3) ones(1,2*M3-mod11states(i)-1)];
        end
end
% display and record each modulus' thermometer code and dynamic range
diary
disp('index_No. dynamic_range thermo_code of mod  9')
disp([index' Drange' code1])
disp('index_No. dynamic_range thermo_code of mod 10')
disp([index' Drange' code2])
disp('index_No. dynamic_range thermo_code of mod 11')
disp([index' Drange' code3])
% finally we get output voltage related to the index No.
mod9Vout = sin(pi*index/pt_cycle9).^2';
mod10Vout =sin(pi*index/pt_cycle10).^2';
mod11Vout = sin(pi*index/pt_cycle11).^2';
disp('simulated output signal of each modulus are as follows')
disp(' index_No dynamic_range mod9 out mod10 out mod11 out')
disp([index' Drange' mod9Vout mod10Vout mod11Vout])
```

```
diary off
```

D. idlprty.m

```
% Thesis program No.4(idlprty.m)
% This m-file generates the threshold value of each parity circuit in the number of
% modulus with the least significant bit (LSB) The number of modulus in LSB and
% the desired percentage of decimation is required to get the threshold voltage.
% This program is designed for 8-bit SNS ADC LabVIEW circuit
%
%                                                    Hiromichi Yamakoshi
% Thesis  project.                    Jul 27 1994
clc,clg                              % Input the required information
m=input('Please input the number of the least significant mod. :')
n=input('And input the desired  percent of the decimation width :')
Drange = m*(0:1/m:1);
Vin =pi*Drange/(2*m);
Vth = sin(pi*Drange/(2*m)).^2;       % First, get threshold value of mod.m
dVin=Vin(2)-Vin(1);                  % get d(Vin) that is used to get %decimation
dtmod=n*dVin/200;                    % get ±0.5*n(%) of d(Vin) from Vth
t1=Vin-dtmod;
t2=Vin+dtmod;
tmod=[t1;t2];                              % create parity check
tmod=reshape(tmod,1,2*length(Vin));        % input value and that
tmod=tmod(:,2:length(tmod)-1);             % element vector
PVth=sin(tmod).^2;
plot(tmod,PVth,tmod,PVth,'x')        % compute and graphically display the
set(gca,'XTickLabels',[])            % threshold value of parity circuit
hold on
Vthline=Vth'*ones(1,length(Vin));
tolanceline=PVth'*ones(1,length(Vin));
        for i=1:length(Vin)
plot(Vin,Vthline(i,:),'c-.',Vin,tolanceline,'r-')
```

```
end
hold off
PVth'
```

E. sys_simu.m

```
% Thesis program No.5(sys_simu.m)
% This m-file generates the number of folds and thermometer code of each sample
% point of each modulus, the number of sample point between two thresholds of
% each comparator, and predicted normalized LabVIEW output of each sample
% point. In addition to these, this m-file also returns the closest threshold level and
% its difference for each sampling point, and parity bit information. The desired
%decimation information is required to get these value above.
% This program is designed for 8-bit SNS ADC LabVIEW circuit
%
%                                                    Hiromichi Yamakoshi

% enter the desired %tolerance
n=input('Input the desired %decimation : ');

% First, we have to set up LAB VIEW setting information.

total_index = 5000;              % 5000 sample s is used as the total index.
pt_cycle9  = total_index/110;    % 110,99,and 90 are used as number of cycle of
pt_cycle10 = total_index/99;     % each modulus,and generates the number of
pt_cycle11 = total_index/90;     % sample point per cycle for each moduli.
                                 %       using the info above, we
pt_state =  pt_cycle9/(2*9);     %       get the # of point per
                                 %       threshold state for each mod.
%
% now input the # of sample points of observation. Now we choose modulus
% 9,10,and 11 for 8-bit SNS ADC. We use 256 dynamic range states out of 990
```

```
% So we actually need the index of sample number, 0 to 646.
%
index=0:646;
mod9folds  = fix(index/pt_cycle9);          % get the #of folding
mod10folds = fix(index/pt_cycle10);         % for each moduli
mod11folds = fix(index/pt_cycle11);         % related to index

% Next we number the ststes of each modulus. each mod. has the
% number which index# from 0 to 2*(#of mod) -1 in one cycle
% (Ex.) they have 18 states in mod 9. (mod 9 has 9 states)

mod9states  = fix((index-(mod9folds*pt_cycle9))/pt_state);
mod10states = fix((index-(mod10folds*pt_cycle10))/pt_state);
mod11states = fix((index-(mod11folds*pt_cycle11))/pt_state);

Drange = mod9folds*(2*9) + mod9states;     % get related dynamic range

%       Next, we create the thermometer code-expression of
%       each modulus (i,e, 001111111 for mod 9, and so on)
M1=9;, M2=10;, M3=11;
code1=zeros(length(mod9states),M1-1);
code2=zeros(length(mod10states),M2-1);
code3=zeros(length(mod11states),M3-1);
% for mod 9
for i=1:length(mod9states);
        if mod9states(i) < M1
          code1(i,:)=[zeros(1,M1-mod9states(i)-1) ones(1,mod9states(i))];
        else            %if mod9states(i) >= M1
          code1(i,:)=[zeros(1,mod9states(i)-M1) ones(1,2*M1-mod9states(i)-1)];
        end
end
% for mod 10
```

```
for i=1:length(mod10states);
        if mod10states(i) < M2
         code2(i,:)=[zeros(1,M2-mod10states(i)-1) ones(1,mod10states(i))];
        else
         code2(i,:)=[zeros(1,mod10states(i)-M2) ones(1,2*M2-mod10states(i)-1)];
        end
end
% for mod 11
for i=1:length(mod11states);
        if mod11states(i) < M3
         code3(i,:)=[zeros(1,M3-mod11states(i)-1) ones(1,mod11states(i))];
        else
         code3(i,:)=[zeros(1,mod11states(i)-M3) ones(1,2*M3-mod11states(i)-1)];
        end
end
% display and record each modulus' thermometer code and dynamic range
diary
disp('index_No. dynamic range   thermo_code of mod 9')
disp([index' Drange' code1])
disp('index_No. dynamic_range thermo_code of mod 10')
disp([index' Drange' code2])
disp('index_No. dynamic_range thermo_code of mod 11')
disp([index' Drange' code3])
diary off

% Now we get output value related to the index No.
%
mod9Vout = sin(pi*index/pt_cycle9).^2;
mod10Vout= sin(pi*index/pt_cycle10).^2;
mod11Vout= sin(pi*index/pt_cycle11).^2;

diary
```

```
disp(['index_No. Vin_ index mod9out mod10out mod11out'])
disp([index' Drange' mod9Vout' mod10Vout' mod11Vout'])
diary off

% then we get the difference between two threshold
% and threshold value of each mod. (for one cycle)
%
t1=linspace(0,pi,2*M1 +1);,Vth1=sin(t1).^2;dVth1=diff(Vth1); % get threshold &
t2=linspace(0,pi,2*M2 +1);,Vth2=sin(t2).^2;dVth2=diff(Vth2); % its diff. for
t3=linspace(0,pi,2*M3 +1);,Vth3=sin(t3).^2;dVth3=diff(Vth3); % each modulus.
diary
disp(['threshold value of mod 9'])
disp(Vth1')
disp(['threshold value of mod 10'])
disp(Vth2')
disp(['threshold value of mod 11'])
disp(Vth3')
disp(['the difference between two threshold of mod 9'])
dVth1'
disp(['the difference between two threshold of mod 10'])
dVth2'
disp(['the difference between two threshold of mod 11'])
dVth3'
diary off
clear t1 t2 t3

% Begin to get the closest threshold value as for each sample point.
% after the following process we get the closest threshold value and
% its difference for every sample point.

Adiff9Vout =zeros(1,length(index));,Bdiff9Vout=zeros(1,length(index));
Adiff10Vout=zeros(1,length(index));,Bdiff10Vout =zeros(1,length(index));
```

```
Adiff11Vout=zeros(1,length(index));,Bdiff11Vout=zeros(1,length(index));
diff9Vout =zeros(1,length(index));,closeVth9 = mod9states;
diff10Vout=zeros(1,length(index));,closeVth10 = mod10states;
diff11Vout=zeros(1,length(index));,closeVth11 = mod11states;
%
% A: get difference between Vout@index pt. & just below Vth (for positive slope),
% or between Vout@index pt. & just above Vth (for negative slope)
% B: get difference between Vout@index pt. & just above Vth (for positive slope),
% or between Vout@index pt. & just below Vth (for negative slope)

for i=1:length(index)
        Adiff9Vout(i) =mod9Vout(i) -Vth1(mod9states(i)+1);
        Adiff10Vout(i)=mod10Vout(i)-Vth2(mod10states(i)+1);
        Adiff11Vout(i)=mod11Vout(i)-Vth3(mod11states(i)+1);
        Bdiff9Vout(i) =mod9Vout(i) -Vth1(mod9states(i)+2);
        Bdiff10Vout(i)=mod10Vout(i)-Vth2(mod10states(i)+2);
        Bdiff11Vout(i)=mod11Vout(i)-Vth3(mod11states(i)+2);
end
%
% Compare A & B, and show the diff.value from which Vth's stage.
%
for i=1:length(index)
        if abs(Adiff9Vout(i)) < abs(Bdiff9Vout(i))
                diff9Vout(i) = Adiff9Vout(i);
        else
                diff9Vout(i) = Bdiff9Vout(i);
                closeVth9(i) =mod9states(i)+1;
        end
        if abs(Adiff10Vout(i)) < abs(Bdiff10Vout(i))
                diff10Vout(i) = Adiff10Vout(i);
        else
                diff10Vout(i) = Bdiff10Vout(i);
```

```
            closeVth10(i) =mod10states(i)+1;
        end
        if abs(Adiff11Vout(i)) < abs(Bdiff11Vout(i))
            diff11Vout(i) = Adiff11Vout(i);
        else
            diff11Vout(i) = Bdiff11Vout(i);
            closeVth11(i) =mod11states(i)+1;
        end
end
% improve the effect of very top/bottom threshold case
% (on this MATLAB code, I treat the normalized "0"/"1"
%       as a threshold, but actually there are not.)

% for mod 9
for i=1:length(index)
        if mod9states(i) == 0;
            diff9Vout(i) = Bdiff9Vout(i);
            closeVth9(i) = 1;
        elseif mod9states(i)== 9;
            diff9Vout(i) = Bdiff9Vout(i);
            closeVth9(i) = 8;
        elseif mod9states(i) == 8;
            diff9Vout(i) = Adiff9Vout(i);
            closeVth9(i) = 8;
        elseif mod9states(i) ==17;
            diff9Vout(i) = Adiff9Vout(i);
            closeVth9(i) = 1;
        elseif closeVth9(i) ==10;
            closeVth9(i) = 8;
        elseif closeVth9(i) ==11;
            closeVth9(i) = 7;
        elseif closeVth9(i) ==12;
```

```
            closeVth9(i) = 6;
        elseif closeVth9(i) ==13;
            closeVth9(i) = 5;
        elseif closeVth9(i) ==14;
            closeVth9(i) = 4;
        elseif closeVth9(i) ==15;
            closeVth9(i) = 3;
        elseif closeVth9(i) ==16;
            closeVth9(i) = 2;
        elseif closeVth9(i) ==17;
            closeVth9(i) = 1;
        end
end

%for mod10
for i=1:length(index)
        if mod10states(i) == 0;
            diff10Vout(i) = Bdiff10Vout(i);
            closeVth10(i) = 1;
        elseif mod10states(i)== 10;
            diff10Vout(i) = Bdiff10Vout(i);
            closeVth10(i) = 9;
        elseif mod10states(i) == 9;
            diff10Vout(i) = Adiff10Vout(i);
            closeVth10(i) = 9;
        elseif mod10states(i) ==19;
            diff10Vout(i) = Adiff10Vout(i);
            closeVth10(i) = 1;
        elseif closeVth10(i) ==11;
            closeVth10(i) = 9;
        elseif closeVth10(i) ==12;
            closeVth10(i) = 8;
```

```
        elseif closeVth10(i) ==13;
                closeVth10(i) = 7;
        elseif closeVth10(i) ==14;
                closeVth10(i) = 6;
        elseif closeVth10(i) ==15;
                closeVth10(i) = 5;
        elseif closeVth10(i) ==16;
                closeVth10(i) = 4;
        elseif closeVth10(i) ==17;
                closeVth10(i) = 3;
        elseif closeVth10(i) ==18;
                closeVth10(i) = 2;
        elseif closeVth10(i) ==19;
                closeVth10(i) = 1;
        end
end

%for mod11
for i=1:length(index)
        if mod11states(i) == 0;
                diff11Vout(i) = Bdiff11Vout(i);
                closeVth11(i) = 1;
        elseif mod11states(i)== 11;
                diff11Vout(i) = Bdiff11Vout(i);
                closeVth11(i) = 10;
        elseif mod11states(i) ==10;
                diff11Vout(i) = Adiff11Vout(i);
                closeVth11(i) = 10;
        elseif mod11states(i) ==21;
                diff11Vout(i) = Adiff11Vout(i);
                closeVth11(i) = 1;
        elseif closeVth11(i) ==12;
```

```
            closeVth11(i) = 10;
        elseif closeVth11(i) ==13;
            closeVth11(i) = 9;
        elseif closeVth11(i) ==14;
            closeVth11(i) = 8;
        elseif closeVth11(i) ==15;
            closeVth11(i) = 7;
        elseif closeVth11(i) ==16;
            closeVth11(i) = 6;
        elseif closeVth11(i) ==17;
            closeVth11(i) = 5;
        elseif closeVth11(i) ==18;
            closeVth11(i) = 4;
        elseif closeVth11(i) ==19;
            closeVth11(i) = 3;
        elseif closeVth11(i) ==20;
            closeVth11(i) = 2;
        elseif closeVth11(i) ==21;
            closeVth11(i) = 1;
        end
end
diary
disp([index Drange' diff9Vout' closeVth9' diff10Vout' closeVth10' diff11Vout'
closeVth11'])
diary off
clear Adiff9Vout Adiff10Vout Adiff11Vout Bdiff9Vout Bdiff10Vout Bdiff11Vout

% and finally, create parity circuit portion
t = linspace(0,pi/2,M1+1);
dt=t(2)-t(1);
dtmod=n*dt/200
tlow=t-dtmod;, tup=t+dtmod;
```

```
tmod=[tlow;tup];,tmod=reshape(tmod,1,2*length(t));          %create the threshold
tmod=tmod(:,2:length(tmod)-1);                              % value of parity
Vparity=sin(tmod).^2;
Vparity=reshape(Vparity,2,M1);
Vprtyup=Vparity(1,:);,Vprtydn=Vparity(2,:);
Vprtypupdiff=Vprtyup-Vth1(:,1:9);,Vprtypupdiff=Vprtypupdiff(2:9);
Vprtypdndiff=Vprtydn-Vth1(:,1:9);,Vprtypdndiff=Vprtypdndiff(2:9);
Vprtynupdiff=Vth1(:,2:10)-Vprtyup;,Vprtynupdiff=Vprtynupdiff(1:8);
Vprtyndndiff=Vth1(:,2:10)-Vprtydn;,Vprtyndndiff=Vprtyndndiff(1:8);
prtyind=zeros(1,length(index));
for i=1:length(index);                      %create parity bit information
        if diff9Vout(i) < 0
                if abs(diff9Vout(i)) > abs(Vprtyndndiff(closeVth9(i))) & ...
                  abs(diff9Vout(i)) < abs(Vprtynupdiff(closeVth9(i)))
                                prtyind(i)=0;
                else
                                prtyind(i)=1;
                end
        else
                if abs(diff9Vout(i)) > abs(Vprtypupdiff(closeVth9(i))) & ...
                  abs(diff9Vout(i)) < abs(Vprtypdndiff(closeVth9(i)))
                                prtyind(i)=0;
                else
                                prtyind(i)=1;
                end
        end
end
diary
disp([index' Drange' diff9Vout' closeVth9' prtyind'])
diary off
```

LIST OF REFERENCES

1. Paul Horowitz and Winfield Hill, *The Art of Electronics Second Edition,* Cambridge University Press, New York, NY, 1989.

2. William D. Stanley, *Operational Amplifiers with Linear Integrated Circuits Third Edition*, Merill, New York, NY, 1994.

3. Jorge A. Esparza, "An Analog Processing Architecture for High-Speed Analog Digital Conversion", Master's Thesis, Naval Postgraduate School, Monterey, CA, December, 1993.

4. A. Arbel and R. Kurz, "Fast ADC," *IEEE Trans. on Nuclear Science*, Vol. NS-22, February 1975.

5. Udo Fiedler and Dieter Seitzer, "A High-Speed 8 Bit A/D Converter Based on a Gray-Code Multiple Folding Circuit," *IEEE J. of Solid-state Circuits*, Vol. SC-14, No.3, June 1979.

6. Johan van Valburg and Rudy J. van de Plassche, "An 8-b 650-MHz Folding ADC," *IEEE J of Solid-state Circuits*, Vol. 27, No. 12, December 1992.

7. H. F. Taylor, "An Optical Analog to Digital Converter - Design and Analysis," *IEEE J. of Quantum Electronics,* Vol. QE-15, pp. 210-216, 1975.

8. Henry F. Taylor, M.J. Taylor, and P.W. Bauer, "Electro-Optic Analog-to-Digital Conversion Using Channel Waveguide Modulators," *Applied Physics Letters*, pp.559-561 Vol. 32 No. 9 MAY 1978.

9. G. D. H. King and R. Cebulski, "Analogue to Digital Conversion Using Integrated Electro-Optical Interferometers," *Electronic Letters,* Vol. 18, pp. 1099 - 1100, 1982.

10. Richard A. Becker et, al, "Wide-Band Electrooptic Guided-Wave Analog to Digital Converters," *Proc. IEEE,* Vol. 72, pp. 802 - 818, 1984.

11. Pace, P. E., and Styer, D., "High Resolution Encoding Process for an Integrated Optical Analog-to-Digital Converter," *Optical Engineering*, Vol. 33, pp. 2638-2645, 1994.

12. R. J. Pieper, P. E. Pace, J. P. Powers, R. Van de Viere, C. C. Foster, R. Walley and H. Yamakoshi, "Feasibility Demonstration of a High-Resolution Integrated Optical Analog-to-Digital Converter," PSAA-IV, 4th Annual ARPA Symp. on Photonic Systems for Antenna Applications, Naval Postgraduate School, Monterey, CA, January 1994.

13. P. E. Pace, J. P. Powers, R. J. Pieper, R. Walley, H. Yamakoshi, C. Crowe, and B. Nimri, "8-Bit Integrated Optical SNS ADC," Proceedings of the Twenty-Seventh Southeastern Symposium of System Theory, Mississippi State University, March 1995.

14. Amnon Yariv, *Optical Electronics Fourth Edition*, Saunders College Publishing, Philadelphia, Pa, 1991.

15. Hiromichi Yamakoshi, "The Experimental Circuit of Analog to Digital Circuit Using Symmetrical Number System Scheme," EC2990 Project, Naval Postgraduate School, March 1994.

16. Jeff Benson, "Simulation of Three Optical Interferometer Outputs," EC2990 Project, Naval Postgraduate School, June 1994.

17. Craig A. Crowe "Optical SNS Folding Circuit Design," Master's Thesis, Naval Postgraduate School, Monterey, CA, March 1995.

18. Broadband Communications Products, Inc., *Operators Manual Model 400 Laser Transmitter*, Broadband Communications Products, Inc., Melbourne, FL, 1993.

19. New Focus, Inc., *Models 1801 and 1811 User's Manual*, New Focus, Inc., Sunnyvale, CA, 1994.

20. Analog Devices Inc., *Amplifier Reference Manual*, Analog Devices Inc., Norwood, MA, 1992.

21. Data Delay Devices, Inc., Product Catalog No.94, Data Delay Devices, Inc., Clifton, NJ, 1994.

22. Tektronix, Inc.,*TDS 420& TDS 460 Digitizing Oscilloscopes User Manual*, Tektronix Inc., Beaverton, OR, 1993.

23. Hewlett Packard Co., HP16500B/16501A Logic Analysis System User's Reference, Hewlett Packard Co., Colorado Springs, CO, 1993.

BIBLIOGRAPHY

1. J. P. Powers, *An Introduction to Fiber Optic Systems*, Richard D. Irwin, Inc., and Aksen Associates, Homewood, IL, 1993.

2. M. Morris Mano, *Digital Design Second Edition*, Prentice Hall, Englewood, NJ, 1991.

3. Robert F. Coughlin, Frederick F. Driscoll, *Operational Amplifiers & Linear Integrated Circuits Fourth Edition*, Prentice Hall, Englewood, NJ, 1991.

4. Henry W. Ott, *Noise Reduction Techniques In Electronic Systems Second Edition*, John Wiley & Sons, Inc., NewYork, NY, 1988.

5. Hiroshi Nishihara, Masamitsu Haruna, Toshiaki Suhara, *Optical Integrated Circuits*, McGraw-Hill, NewYork, NY, 1989.

6. Behzad Razavi, *Principles of Data Conversion System Design*, IEEE Press, Piscataway, NJ, 1995.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Dudley Knox Library, Code 52 Naval Postgraduate School Monterey, California 93943-5101	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	1
4.	Professor Phillip E. Pace, Code EC/Pc Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	3
5.	Professor Ronald J. Pieper, Code EC/Pr Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	2
6.	Professor John P. Powers, Code EC/Po Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	1
7.	Professor Randy L. Borchardt, Code EC/Bt Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5121	1
8.	LT. Hiromichi Yamakoshi, JMSDF 22-24 Umegaoka Aoba-ku Yokohama, Kanagawa, 227 Japan	2
9.	Space and Naval Warfare System Command Department of the Navy PMW-178 Attn: CAPT. Ristorcelli Washington, DC 20363-5100	1

10. Mr. Gerald Peake 1
2301 South Jefferson Davis Hwy
#523
Arlington, VA 22202

11. Richard Patterson, Code 772 1
Nccosc-NRAD division
San Diego, California 92152

12. Rome Laboratory/IRAP 1
Attn: Bill Ziesenitz
32 Hanger Road
Griffiss AFB, NY 13441-4114

13. Rome Laboratory 1
Attn: Capt. Rice
32 Hanger Road, Bldg. 240
Griffiss AFB, NY 13441-4114

14. U. S. Army CECOM-RDEC 1
C3I Acquisition Center
Contract Operations VHFS Office
AMSEL-ACVF-C-AJ (Stop 42)
Attn: Mr. Tom Tuma
Vint Hill Farms Station
Warrenton, VA 22186-5172

15. WL/AAWP-1 1
Attn: Dave Sharpin
Hangar 4B
3050 C Street
Wright-Patterson AFB
Dayton, OH 45433-7300